Universidad
Carlos III de Madrid

Ingeniería Técnica en informática de Gestión

Proyecto Fin de Carrera

Q-learning determinista.

Análisis de la variable Gamma

Autor: Manuel Monge Hernaiz

Tutor: Profesor Dr. D. Daniel Borrajo Millán

Fecha: Octubre 2015

AGRADECIMIENTOS

Al departamento de Informática de la universidad Carlos III por:

“rescatarme tras tanto tiempo”.

Y a Daniel Borrajo por hacerme tan fácil el rescate.

Y por supuesto a todo mi familia ascendente, descendente y lateral.

INDICE

1. INTRODUCION

En el siguiente Proyecto de fin de carrera (PFC) se analiza el comportamiento de la variable gamma en un método de aprendizaje automático Q-learning en su versión determinista.

El objetivo es determinar el comportamiento de este método en función de la dificultad del problema a resolver. Analizando a posteriori los resultados obtenidos para intentar esclarecer si hay alguna relación entre la dificultad y el valor de la variable gamma. Este PFC está orientado al concepto de juegos, concretamente juegos en los que se ha de encontrar un tesoro.

Para ello se ha realizado un programa implementando este método en un mapa de 20x20 casillas. El objetivo del programa es encontrar el tesoro del mapa tantas veces como sea posible. Con este fin deberá explorar y aprender a moverse en las 4 direcciones posibles de las casillas (norte, sur, este y oeste) evitando a los enemigos dispuestos en el mapa. Dichos enemigos se distribuirán en el mapa en función de la dificultad deseada. La dificultad ira desde fácil hasta difícil.

Para ilustrar al lector sobre el método, en la sección 2 se va a mostrar el estado de la cuestión. Explicare en qué consiste el método Q-learning en su versión determinista.

En la sección 3 se explicarán los objetivos de este PFC, describiendo que se quiere obtener desarrollando este PFC. A continuación en la sección 4 se escribirá una memoria completa de las tareas llevadas a cabo, la arquitectura del programa y la interconexión entre sus distintas funciones. También se establecerá una descripción de alto nivel, un manual de usuario y un manual de referencia.

En la sección 5 se han descrito 3 subsecciones una para cada caso probado. Estos casos son fácil, medio y difícil, en cada uno de los casos se ha realizado pruebas con 3 valores diferentes de la variable gamma. Para cada uno de estos valores se mostrará el mapa con las casillas exploradas junto con una gráfica con el número estados recorridos por cada ciclo.

Se mostrara en la sección 5 una descripción completa de las pruebas realizadas. Además serán analizados los resultados obtenidos en la sección 6. Posteriormente se establecerá una conclusión sobre todo el trabajo realizado en la sección 7. Para concluir en la sección 8 se describirán las líneas futuras, esbozando ideas generales sobre la continuación de este análisis y sus posibles ampliaciones. Finalmente y como último paso se añadirá la bibliografía consultada en la sección 9.

2. ESTADO DEL ARTE

2.1. APRENDIZAJE POR REFUERZO

Para entender el aprendizaje por refuerzo citare a Fernando Fernández Rebollo. Tesis Doctoral. 2002. Universidad Carlos III de Madrid: "El aprendizaje por refuerzo consiste en aprender a decidir, ante una situación determinada, qué acción es la más adecuada para lograr un objetivo".

Es decir el aprendizaje por refuerzo es un modelo basado en la automatización y la interacción con el entorno. Con este fin posee una serie de estados y unas determinadas acciones. En él siempre existe un agente que avanza por prueba/error dentro de un entorno. Este avance está guiado (reforzado) por unos valores de refuerzo o ganancia que son proporcionados por el propio entorno.

En resumen es un método de aprendizaje que determina cuál es la mejor acción en cada estado. Existe una función de recompensa que puede ser aplicada a cada una de las acciones del estado en el que nos encontramos.

Gracias a este sistema de recompensa, el método recibe un refuerzo que alimenta el conocimiento que se tiene del entorno. Esta es la forma en que el método aprende por un refuerzo tanto si es positivo como si es negativo. Por lo tanto el aprendizaje es por prueba y error.

Una de las particularidades del aprendizaje por refuerzo es que trata de buscar el equilibrio entre la exploración del entorno, obteniendo así valores nuevos con los que aprender y la explotación de los mismos, es decir el conocimiento actual que tiene del entorno. En el caso de este PFC la implementación de este método se ha desarrollado mediante la técnica conocida como Q-learning en su versión determinista.

2.2. Q-LEARNING

Q-learning es una técnica que utiliza el concepto de aprendizaje por refuerzo. La tecnica está basada en la utilización de un agente que será el encargado de la toma de decisiones. El agente necesita una política que le indique en un determinado estado las acciones a ejecutar. Es decir para un determinado estado cada una de las posibles acciones es evaluada por una política que decide cuál es la mejor acción. Una vez elegida la mejor acción el sistema es alimentado con un refuerzo correspondiente a la acción para ese estado determinado.

Normalmente se tiene una matriz Q que tiene por filas todos los posibles estados de un problema concreto y como columnas todas las acciones que pueden llevarse a cabo para cambiar de estado.

El proceso es el siguiente: el agente se encuentra en un estado e usando la política de evaluación decide una acción a que le lleva a un nuevo estado e'. Entonces se utiliza el refuerzo del estado e para la acción a para actualizar utilizando la siguiente formula:

$$Q_t(e,a) \leftarrow r(e,a) + \gamma \max_{b \in A} Q_{t-1}(e',b)$$

Posteriormente el agente recorre los distintos estados buscando la mejor acción. Cada vez que cambiamos de estado mediante una acción, hay que actualizar la acción correspondiente del estado en el que estamos. En la formula anterior es donde aparece la variable gamma que voy a analizar. La explicación de la fórmula es la siguiente: El valor de la matriz Q que corresponde a un estado y acción, ha de ser actualizado con el refuerzo para ese estado y acción añadiendo el resultado de multiplicar gamma por la mejor acción del valor del siguiente, dicho estado es al que se llega tras tomar la acción evaluada originalmente.

3. OBJETIVO

El objetivo de este PFC es analizar el comportamiento de la variable gamma en una implementación determinista del método de aprendizaje Q-learning. Este método podría traducirse como un valor que indica “la importancia de los futuros estados en la decisión”. El objetivo real de este PFC es analizarlo desde un punto de vista diferente añadiendo el parámetro de dificultad al estudio.

Analizaré el comportamiento de la variable modificando su valor en diversas cantidades, también se analizará esta variación en distintos grados de dificultad, básicamente el grado de dificultad cambiará en el número de enemigos a sortear, así como su posición, buscando aumentar la dificultad con determinados esquemas. Este PFC se lleva a cabo para intentar sacar una conclusión. Intentando saber si un valor alto o bajo de la variable gamma en sí mismo es mejor en unos casos de dificultad que en otros. También se podría concluir que no hay variación en función de la dificultad.

4. MEMORIA

4.1. Introducción

En los siguientes apartados se mostrarán las tareas llevadas a cabo para realizar este PFC y una pequeña introducción sobre las decisiones tomadas y los motivos para tomarlas. El método de Q-learning en el que está basado el programa es el método determinista, dicho método es el más sencillo de todos, esta elección ha sido así porque entiendo que el concepto de dificultad debe probarse con la menor interferencia posible. Se deben utilizar los cálculos más básicos si realmente se desea medir el concepto "dificultad", esta ha sido la motivación para elegir este método.

El proyecto consta de dos grandes partes, la primera de ella es el análisis que establecerá a grandes rasgos los requisitos a la hora de realizar el diseño. La segunda es el diseño en el que se mostrará el diagrama de la aplicación y por último la parte de programación. Esta última no será desarrollada pues los algoritmos de la aplicación son suficientemente sencillos para cualquier programador.

4.2. Arquitectura

4.2.1. DESCRIPCION

Es necesario desarrollar una arquitectura que me permita introducir el concepto de dificultad, he decidido introducir ese concepto en la arquitectura en el concepto de mapa, por lo tanto la aplicación desarrollará su cometido en un mapa finito y de una dificultad determinada.

Por lo tanto la arquitectura del programa está compuesta por los siguientes elementos:

- Ha de implementar una técnica de Q-Learning en su versión determinista.
- Una interfaz gráfica que permita visualizarlo todo.
- Una parte que se encargue de generar datos para su posterior análisis.

La arquitectura constara de un método principal, que es el encargado de todo el peso del programa. El método principal en líneas generales se encargara de desplazar al explorador utilizando la técnica de Q-learning determinista, además inicialmente invocara a los métodos que se encarguen de gestionar el mapa y de gestionar la parte gráfica, para en último lugar explotar los resultados obtenidos.

La interfaz gráfica será sencilla ya que nos interesa solo para determinar visualmente como se desplaza el explorador. La dificultad del mapa dependerá del número de enemigos y la distribución de los mismos. El generador de datos para su posterior análisis ha de ser sencillo pero eficaz mostrando los datos para poder analizarlos a posteriori.

El explorador se moverá con movimientos sencillos en las 4 direcciones cardinales, las decisiones del mismo estarán basadas en la técnica de Q-learning en su versión determinista. El procedimiento Q-learning será estándar y estará compuesto

por una matriz en la que se elegirá la mejor acción, recibirá un refuerzo y actualizará su matriz de datos.

En la figura 1 puede verse un esquema de la arquitectura necesaria para implementar la aplicación:

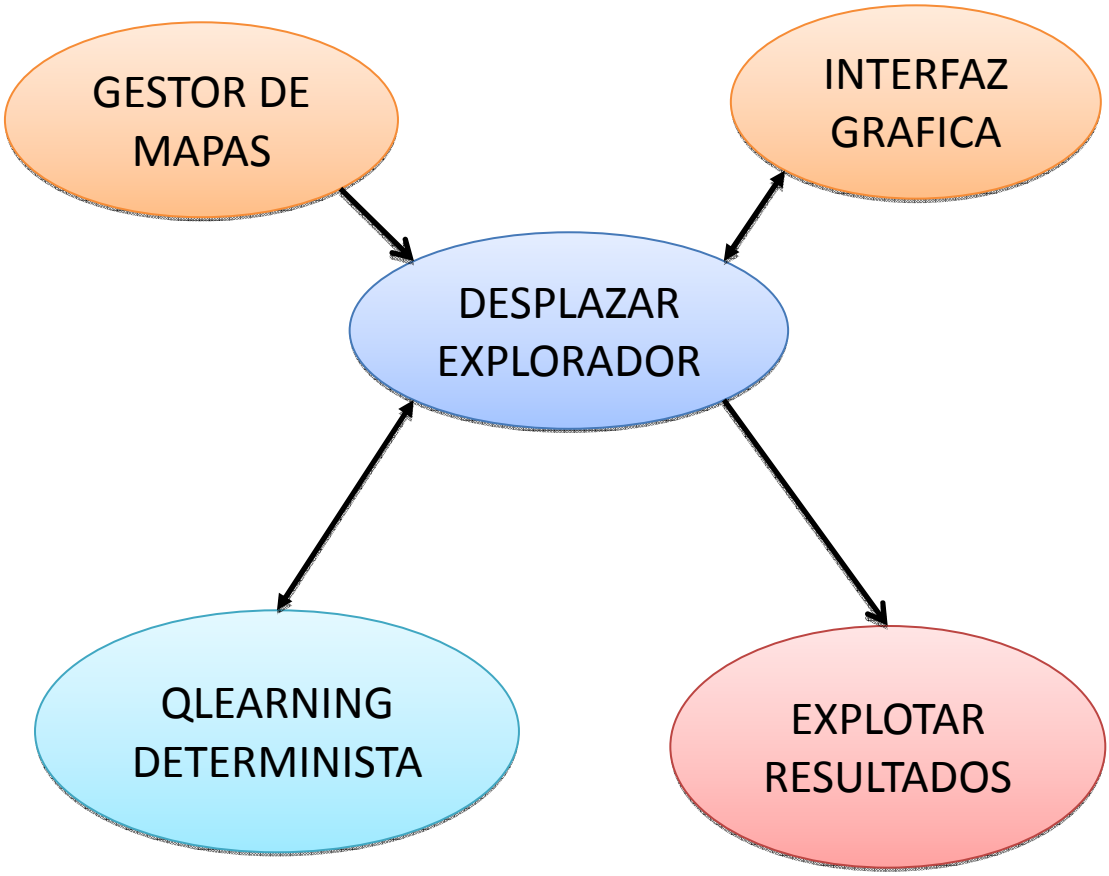


Figura 1 – Arquitectura de la aplicación

4.2.2. DISEÑO

En la figura 2 se muestra una gráfica del diseño de los distintos módulos de la aplicación, en él se pueden observar las llamadas de un método a otro y las variables intercambiadas en cada uno.

El diseño de la aplicación consta de una única clase que invoca al método main, el cual se encarga de crear el objeto de la clase e invocar al método principal, que es el encargado de todo el peso del programa. El método principal en líneas generales, invoca a los métodos que rellenan y pintan la rejilla en pantalla, para posteriormente invocar al método mover explorador tantas veces como ciclos se hayan definido.

El método mover explorador es el encargado de utilizar la técnica Q-Learning para desplazar el explorador por el mapa, para implementar dicha técnica se basa en 3 funciones, mejor acción, recibir refuerzo y actualizar Q-learning estas funciones tienen un nombre auto explicativo.

También existen otro par de métodos en el diseño: resultado final y mejor valor, el primero se encarga de mostrar por pantalla el resultado final compuesto por la matriz de Q-learning y el número tesoros y muertes acaecidas, el segundo se encargar de seleccionar el mejor valor para un determinado estado.

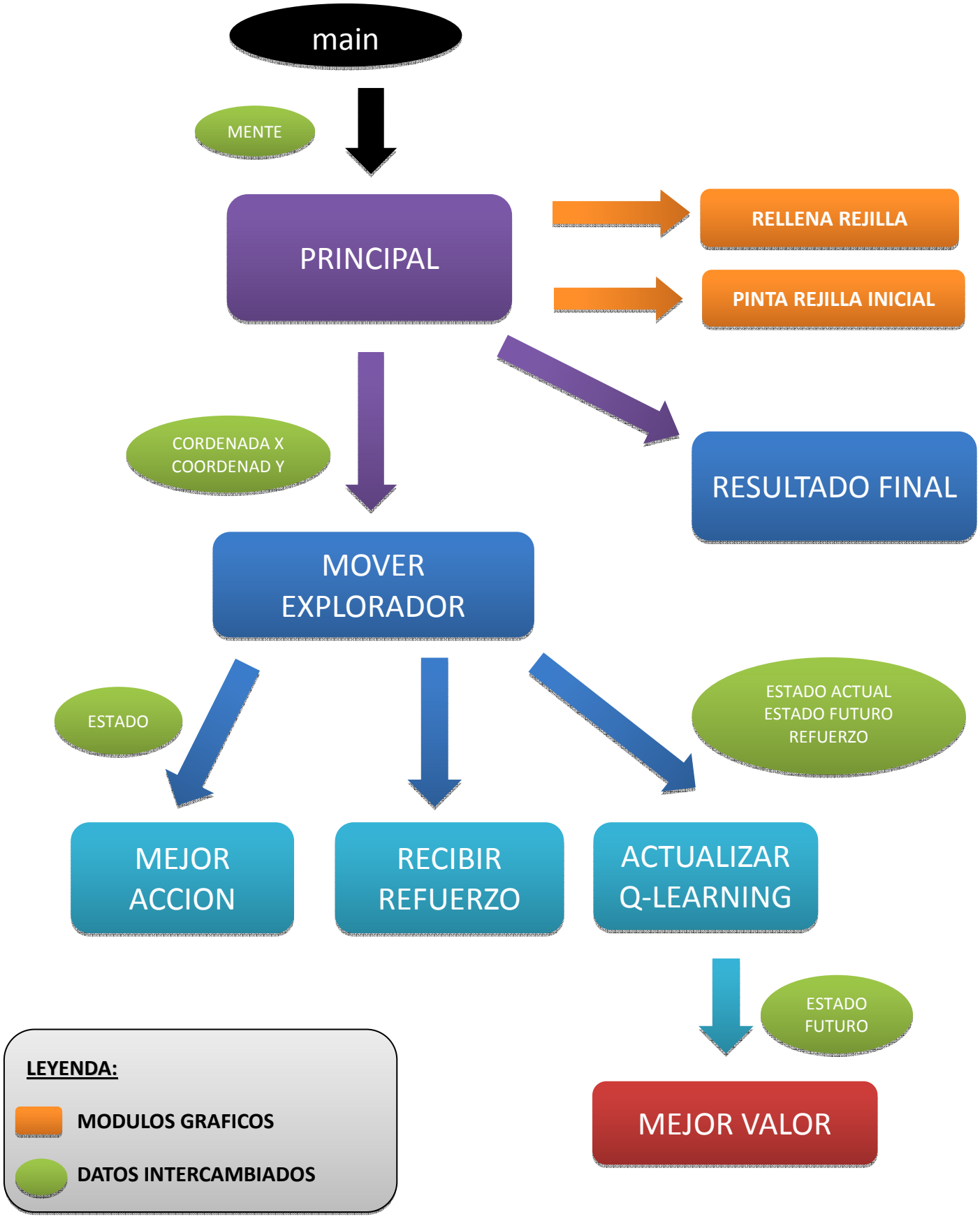


Figura 2 - Diseño de la aplicación.

4.3. Modelo de conocimiento

4.3.1. PRESENTACION

Como ya se ha citado anteriormente el programa consta de una única clase, para que sea fácilmente modificable y ampliable en un futuro. El diagrama de clases que se va a mostrar a continuación no es un diagrama de clases en sí mismo, es un diagrama entre los métodos de la clase. Se puede considerar una ampliación/explicación del diseño de la arquitectura mostrado en la sección anterior.

Se van a comentar los métodos más destacados de la clase, al más alto nivel, con una pequeña explicación de la funcionalidad de los mismos, sin definir entradas y salidas, para ese propósito está escrita la sección 4.4.

Los atributos que modifican cada método también se describirán en la sección 4.4. Esta sección sirva para asentar el modelo de conocimiento, describiendo lo que hace cada método en líneas generales.

4.3.2. METODOS DESTACADOS

Principal:

Es el bucle principal que se encarga de invocar a los distintos métodos.

Rellena Rejilla:

Rellena la variable rejilla del tamaño del mapa con los valores a 0, además de introducir en la rejilla el tesoro y los enemigos.

Pinta Rejilla Inicial:

Pinta en pantalla la rejilla rellena en el apartado anterior por primera vez para poder visualizar los enemigos y el tesoro.

Mover Explorador:

Se encargada de mover el explorador que va recorriendo el mapa, eligiendo el mejor movimiento mediante el método del Q-learning y representarlo en pantalla.

Mejor Acción:

Se encarga de elegir la mejor acción a realizar en un estado concreto, evaluando el mejor valor de la matriz de Q-learning.

Recibir Refuerzo:

Se encarga de recibir un refuerzo evaluando si el explorador está en una casilla de tesoro o de enemigo.

Actualizar Q-learning:

Se encarga de actualizar la matriz de Q-learning, con el valor recibido estableciendo un nuevo valor para la acción de un estado concreto.

Mejor Valor:

Se encarga de evaluar el mejor valor para un estado concreto.

Resultado Final:

Muestra por la consola del sistema el resultado final, tanto la matriz de Q-learning, además indica el número de veces que se obtiene tesoro o que se muere en el intento.

4.4. Descripción de alto nivel

4.4.1. VISTA GENERAL

En esta sección se presenta una vista general de los principales algoritmos que se han desarrollado en el programa, es una simple una enumeración para tener una visión global de los mismos.

El programa consta de cinco algoritmos principales:

- Principal
- Rellena Rejilla
- Mover Explorador
- Recibir Refuerzo
- Actualizar Q-learning

En la siguiente sección se describe cada uno de los algoritmos principales, también se describen las entradas de los mismos, las salidas que producen, los atributos de la clase que modifican así como los submódulos a los que se invocan, además se describirá el pseudocódigo.

Debido al diseño elegido para la clase, varios de los algoritmos (métodos) varían las propiedades de la clase sin recibirlas siquiera como parámetros, este hecho viene motivado por el deseo de la simplicidad y la facilidad de implementación futuras.

4.4.2. PRINCIPALES ALGORITMOS

A continuación se describe en negrita y en azul, los principales métodos, no están todos los que tiene el programa, pero sí los más importantes para poder entender el funcionamiento de la clase.

Principal:

Entradas: Ninguna, obtiene los valores que necesita de las propiedades.

Salidas: El número de estados recorridos por ciclo.

Atributos modificados: Los que modifiquen sus submódulos.

Submódulos: Rellena Rejilla, PintaRejillaIncial, MoverExplorador, ResultadoFinal.

Pseudocódigo:

Invocar método pinta rejilla;

Invocar método pinta rejilla Inicial;

Para el número de ciclos repetir;

Mientras el explorador no muera;

Mover al explorador;

Aumentar el número de estados recorridos

Imprimir el número de estados recorridos;

Rellena Rejilla:

Entradas: Rejilla aunque no es pasada como parámetros.

Salidas: La rejilla rellenado con los valores correspondientes.

Atributos modificados: No modifica atributos de la clase.

Submódulos: No tiene.

Pseudocódigo:

Para todas las columnas;

Par todas las filas;

Poner 0 en la rejilla;

Poner -10 para enemigo y 10 para tesoro en determinadas casillas en función del mapa estudiado.

Mover Explorador:

Entradas: PosicionX y posicionY.

Salidas: RefuerzoRecibido, si es 0 se acaba el ciclo.

Atributos modificados: Modifica la matriz de Q-learning, en sus submétodos.

Submódulos: Mejor acción, RecibirRefuerzo, ActualizarQ-learning.

Pseudocódigo:

Elegir la mejor acción en el estado actual (usando método Mejor acción);

Recibir el refuerzo;

Actualizar la matrizQ;

Devolver el refuerzo (realmente devuelve si hay enemigo o tesoro);

Mejor Accion:

Entradas: Estado.

Salidas: Acción a Realizar.

Atributos modificados: Modifica la variable VecesAleatorio.

Submódulos: Ninguno.

Pseudocódigo:

Elegir un número aleatorio;

Si el numero aleatorio es menor que épsilon (épsilon=valor de aleatoriedad empieza en 90% y se va reduciendo un 10% cada vez que se encuentra el tesoro, hasta un mínimo de 10%);

Recorre todas las acciones de ese estado buscando la mejor acción posible;

En caso contrario se elige aleatoriamente la acción;

Devolver la acción seleccionada;

Recibir Refuerzo:

Entradas: Rejilla aunque no es pasada como parámetros, las propiedades coordenadaX y coordenadaY de la clase aunque no son pasadas como parámetro.

Salidas: El valor del refuerzo recibido.

Atributos modificados: Las propiedades coordenadaX y coordenadaY (no siempre).

Submódulos: No tiene.

Pseudocódigo:

Si encontramos enemigo;

Devolver un refuerzo negativo;

Caso contrario;

Si encontramos tesoro;

Devolver refuerzo positivo;

Reducir la aleatoriedad de la búsqueda. (Un 10%, hasta un mínimo de 10%);

Caso contrario devolver 0;

Devolver refuerzo;

Actualizar Q-learning:

Entradas: Estado actual, estado futuro, refuerzo y acción.

Salidas: La matriz Q-learning modificada.

Atributos modificados: La matriz Q-learning.

Submódulos: No tiene.

Pseudocódigo:

Futuro = Para el estado futuro obtener el máximo valor de acción.

Valor = refuerzo obtenido + gamma multiplicado por el Futuro (obtenido en el paso anterior).

Actualizar MatrizQ (estado, acción) con el valor anterior;

4.5. Manual de usuario

4.5.1. REQUISITOS MINIMOS

Los requisitos mínimos para ejecutar la aplicación son los siguientes:

- Tener instalado java en el sistema operativo correspondiente
- Tener correctamente configurado el PATH del sistema operativo las librerías de java
- Tener correctamente configurado la ubicación de los binarios de java.
- Ser capaz de ejecutar en una ventana de shell el comando:
  java -version
- Dicho comando debe mostrar una salida similar a
  java version "1.8.0_60"
  Java(TM) SE Runtime Environment (build 1.8.0_60-b27)
  Java HotSpot(TM) Client VM (build 25.60-b23, mixed mode, sharing)

En caso de desear modificar el código fuente, será necesario además tener configurados los siguientes parámetros.

- Configurar las librerías de compilación de java
- Configurar correctamente el path con el compilador de java
- Ser capaz de ejecutar en una ventana de shell el comando:
  javac -version
- Dicho comando debe mostrar una salida similar a
  C:\Users\admin>javac -version
  javac 1.8.0_31

4.5.2. MODIFICACION

El programa se puede modificar solo en el código fuente, tanto para variar el número de ciclos, como para variar el valor gamma o alternar entre los distintos casos. Si desea modificarse el número de ciclos bastara con modificar en la ***línea 52*** la variable ciclos. Si se desea alterar el valor de la variable gamma, bastara con modificar en la ***línea 53*** la variable gamma.

Si se desea alternar entre los 3 casos expuestos, bastara con buscar en el código el método ***RellenaRejilla*** comentando y descomentando la parte correspondiente a cada uno de los casos que está especificado en el código fuente.

Cualquier modificación debe ser posteriormente compilada con el compilador java por lo que sería necesario ejecutar en la shell del sistema operativo.

javac Explorador.java

4.5.3. EJECUCION

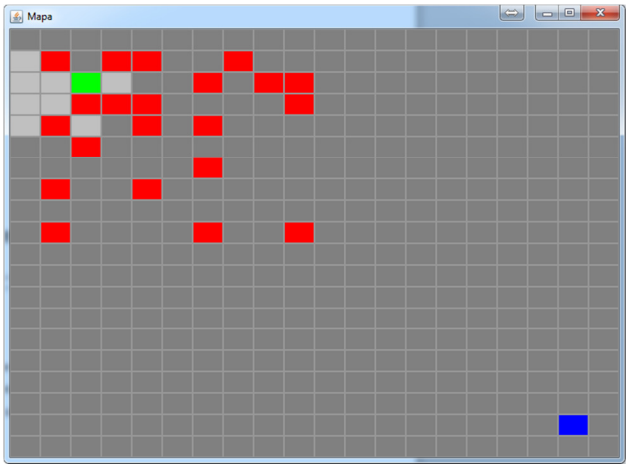
Para ejecutar la aplicación tanto si se ha compilado como si no, bastara con ejecutar en la shell del sistema el siguiente comando (funciona en diversos Sistemas Operativos):

java Explorador

En la ventana la shell se mostrara un único numero por cada línea y superpuesta aparecerá una ventana con 20x20 casillas con una casilla verde en la esquina superior izquierda, tantas casillas rojas como enemigos existan y dentro de las casilla podrá verse avanzar una casilla azul (el explorador en sí mismo).

Tras la ejecución, será necesario esperar un poco, sobre todo cuando el número de ciclos es elevado, cuando se compruebe que la casilla azul no se mueve y se queda fija en las coordenadas 19,19 aparecerán en ***gris oscuro*** las casillas que han sido exploradas y en ***gris claro*** las que nunca fueron exploradas, a partir de ese momento se puede cerrar la aplicación pulsando la X en la esquina superior derecha o pulsando Alt + F4.

Ejemplo:



Posteriormente la ventana de shell recuperara el control, en ella aparecerá el número de estados recorridos por cada ciclo, un numero por cada línea.

4.6. Manual de referencia

La aplicación está programada en java, siguiendo los estándares del lenguaje java, tanto en la implementación de métodos como en la implementación de métodos de clase.

Importa varios componentes estándar de la librería awt y swing, para el manejo gráfico y otras tareas, a continuación un listado de las librerías importadas:

```
import java.io.*;
importjava.awt.GridLayout;
importjava.awt.event.*;
importjava.awt.Color;
importjava.awt.FlowLayout;
importjava.awt.GridLayout;
importjavax.swing.JTextField;
importjavax.swing.JButton;
importjavax.swing.JPanel;
importjavax.swing.JFrame;
importjavax.swing.JLabel;
importjavax.swing.BoxLayout;
importjavax.swing.WindowConstants;
importjava.util.Random;
importjava.util.Random;
```

La aplicación está generada en una única clase con el fin de adaptar el código con facilidad, integrándolo en otro programa más complejo de modo que se pudieran ejecutar varios exploradores como una clase de Q-learning en su versión determinista.

La clase explorador contiene diversos métodos estructurados en dos partes, una parte es la parte gráfica en la que se crea y se muestra una rejilla en pantalla, que será la encargada de representar gráficamente los pasos y movimientos que vayamos generando con el método de Q-learning.

Otra de las partes es la parte del algoritmo de Q-learning, en ella se desarrollan varios métodos para inicializar una matriz Q-learning, así mismo se desarrollan métodos para mover al explorador por la matriz, métodos para evaluar cuál es el

mejor estado dada una posición concreta, un método principal que ejecuta el algoritmo un numero de ciclos determinados en el código del programa.

Se puede cambiar bastante parte de código fuente concretamente la parte grafica es bastante independiente y se podría mejorar, sería una tarea fácil para aquellos programadores que manejen las interfaces graficas de java.

La parte de algoritmo propia de Q-learning también puede modificarse, utilizando la actual como base, pero traería mayores cambios ya que cualquier cambio implicaría cambiar bastante parte del código la única excepción sería quizás cambiar el método de determinista a no determinista. En dicho caso consistiría más bien en añadir más variables al código y complicar un poco más la evaluación de estados para hallar la mejor acción a realizar en cada estado.

5. PRUEBAS

Las pruebas realizadas se detallan en profundidad en esta sección, no obstante ha consistido en una parte importante en el desarrollo de este PFC por lo que se describirá a continuación el proceso.

Las pruebas realizadas se han agrupado en torno al concepto de dificultad propiamente dicho. He establecido tres niveles de dificultad fácil, medio y difícil para cada uno de ellos se ha establecido un caso.

Como concepto de dificultad, se han manejado varios parámetros, dejando siempre fijo un único tesoro en el mapa, variando así otros parámetros. Siendo el parámetro principal el número de enemigos y como secundario la distribución de los mismos en el mapa, de esta manera cuanto más agrupados en una pequeña extensión más fácil y cuanto más *"desplegados en el mapa"* más difícil.

Para el desarrollo de cada caso, se han realizado pruebas de diversos valores de la variable gamma, como se explicará más adelante se han utilizado 3 valores de la misma. Dado que el entorno es determinista y surgida la necesidad de rellenar la matrizQ lo más completamente posible, se ha variado tanto la citada variable gamma para las pruebas, como dinámicamente la variable de épsilon que es básicamente la aleatoriedad con la que se elige la acción.

Nota: *De ahora en adelante en todas la gráficas en las que aparezca el valor de gamma el color del gráfico indicará el valor de gamma: verde para 0,9 granate para el valor 0,7 y verde para el valor 0,5.*

La variable gamma puede describirse como la influencia del refuerzo futuro en el estado actual. Por lo tanto la variable influye en los valores que se insertan en la matrizQ, por lo que su valor alterará la posible acción elegida en futuros ciclos.

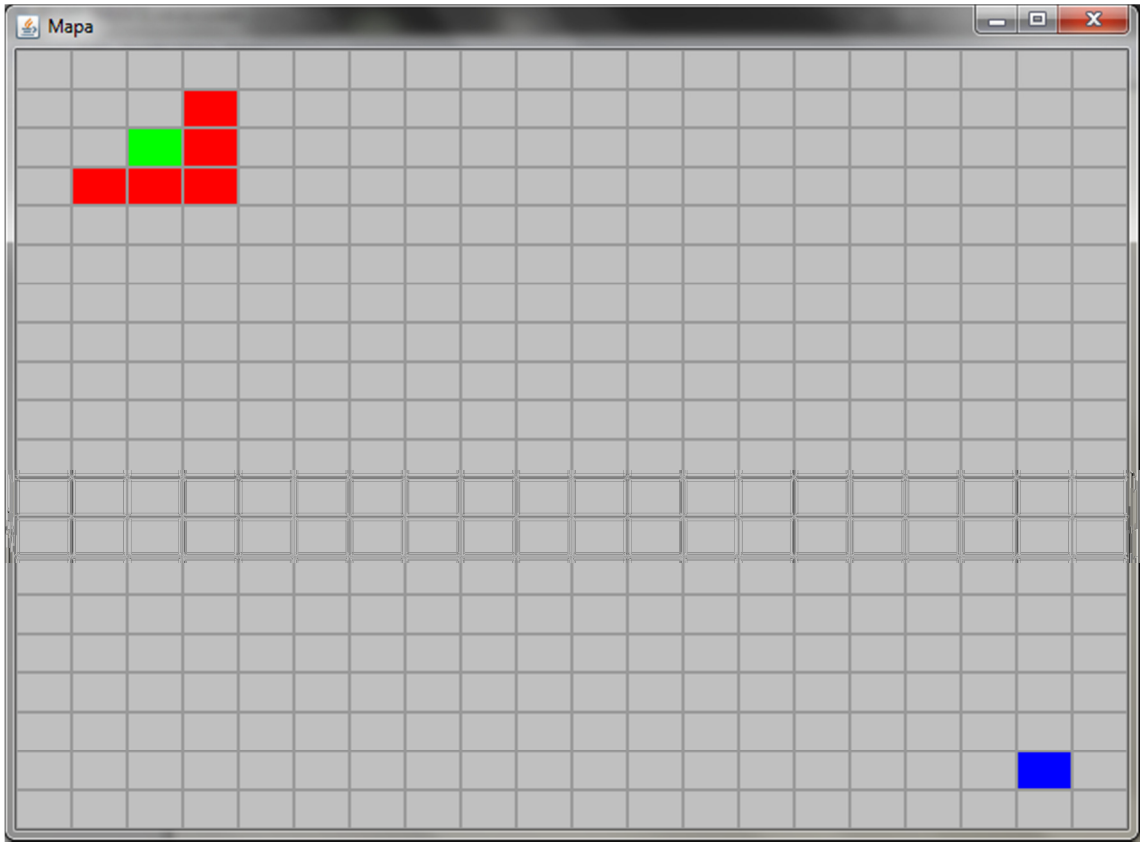
La variable épsilon indica el valor de aleatoriedad que comienza en un 90%, y se va reduciendo un 10% cada vez que se encuentra un tesoro hasta un mínimo del 10%. Por lo tanto esta variable controla la exploración-explotación, a mayor valor más exploración a menor valor más explotación.

Se han realizado diversas pruebas en función de su dificultad, para todas ellas se han tomado 3 valores gamma (0,9 – 0,7 y 0,5) que se mostraran a continuación, las pruebas se han realizado tres veces para cada caso. Se mostraran en color gris claro aquellas casillas que el programa no llega a evaluar.

5.1. Caso1 – Fácil

El caso número uno tiene esta distribución de tesoro y de enemigos como se muestra en la figura 3. Se considera muy fácil debido al escaso número de enemigos en relación al número de casillas como puede apreciarse en la figura 3.

Se toma como medida 1000 ciclos debido a que es un método sencillo y se comporta prácticamente igual después de 100 ciclos.



LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

Figura 3 – Mapa Fácil

5.1.1. GAMMA 0.9

Como puede observarse en la figura 4 el algoritmo explora una parte del mapa y a medida que va reduciéndose épsilon deja de explorar, con este valor de gamma alto se tiene muy en cuenta las recompensas futuras por lo tanto encuentra fácilmente el tesoro. Por ese motivo hay bastante parte del mapa sin explorar.

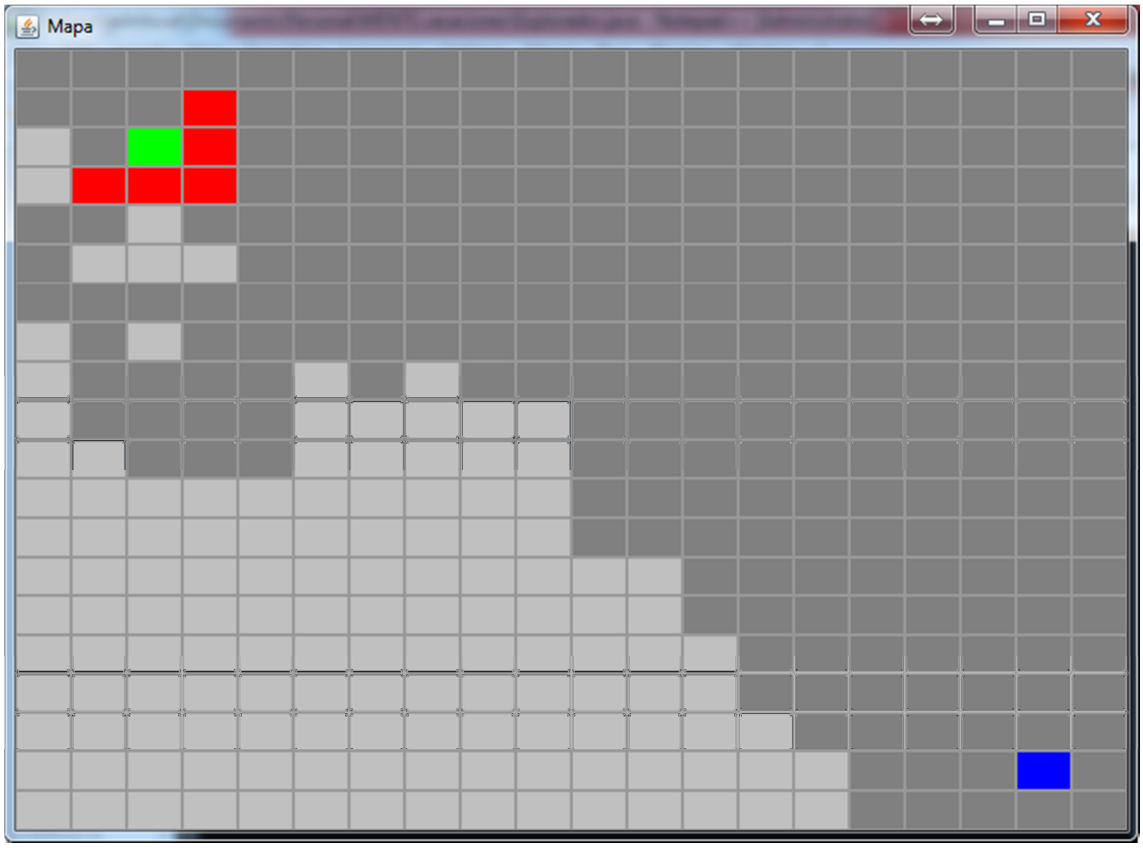


Figura 4- Resultado mapa fácil recorrido con gamma 0.9

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

En la figura 5 se observa la gráfica obtenida al procesar los resultados. En ella se indica el número de estados recorridos en cada ciclo de la aplicación. Como puede observarse a simple vista el número de estados recorridos comienza siendo muy alto. Sin embargo después de un número de ciclos determinados se estabiliza. Esto es porque ha encontrado el tesoro y se reduce épsilon. No obstante el final de ciclo viene marcado por encontrar el tesoro o encontrar un enemigo.

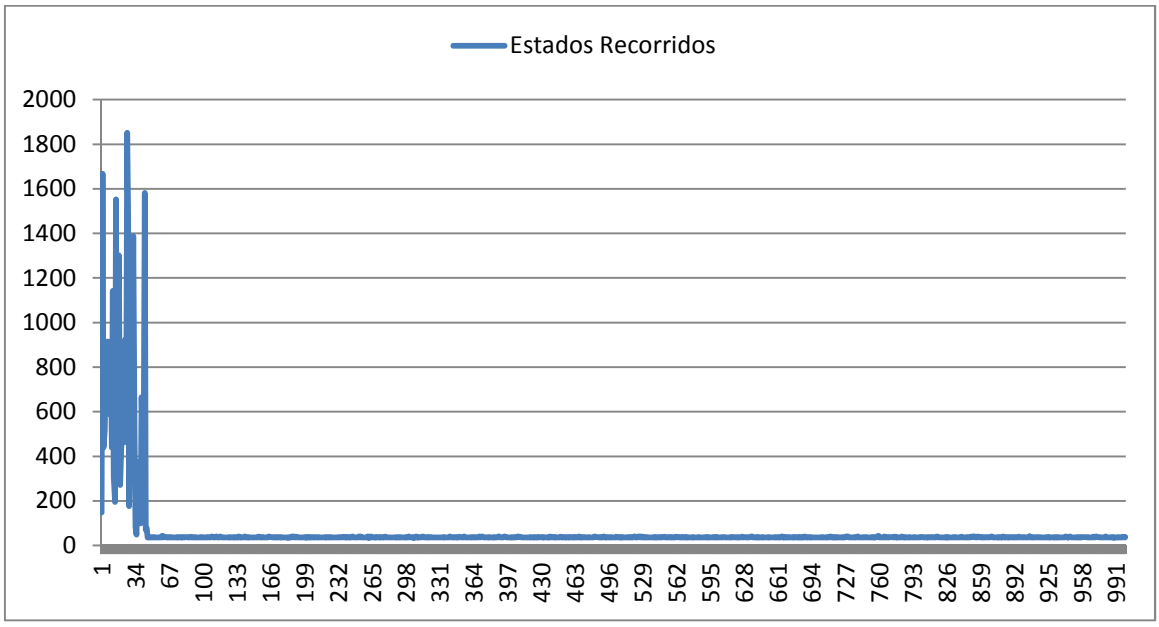


Figura 5 – Número de estados recorridos por ciclo en mapa fácil con gamma 0.9

5.1.2. GAMMA 0,7

Como puede observarse en la figura 6 el algoritmo explora una parte del mapa, con este valor de gamma intermedio se tiene un poco menos en cuenta las recompensas futuras. En este caso el programa se comporta de una manera parecida a la anterior, pronto encuentra el tesoro y al reducir épsilon el valor de explotación de nuevo tiene más peso que la exploración.

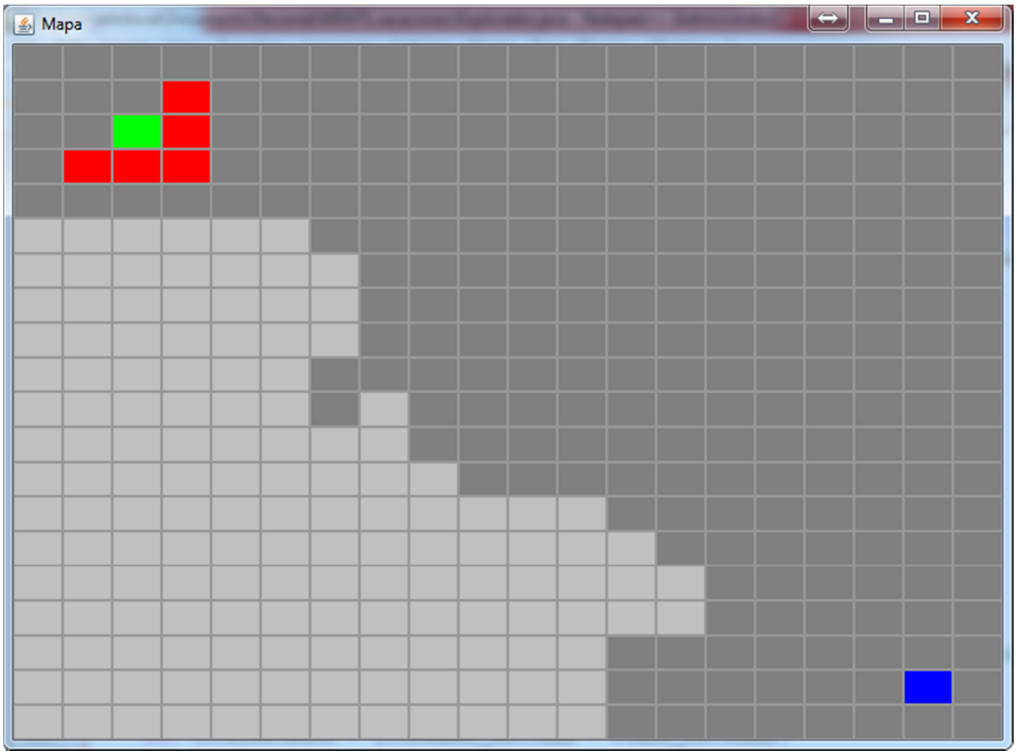


Figura 6 - Resultado mapa fácil recorrido con gamma 0.7

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

En la figura 7 se observa la gráfica obtenida al procesar los resultados, puede observarse que ocurre algo similar al caso anterior el número de ciclos tiende a estabilizarse pronto. Ya que se reduce el épsilon pronto.

Figura 7 - Número de estados recorridos por ciclo en mapa fácil con gamma 0.7

5.1.3. GAMMA 0,5

En la figura 8 se observa que ocurre para este caso de gamma bajo, con este valor se tiene poco en cuenta las recompensas futuras. El programa se comporta de una manera muy parecida a los anteriores casos, a pesar de que debería explorar más, pronto encuentra el tesoro lo que de nuevo reduce el épsilon y hace que tienda más a tener en cuenta el valor de explotación que el de exploración motivo por el que no termina de explorar el mapa.

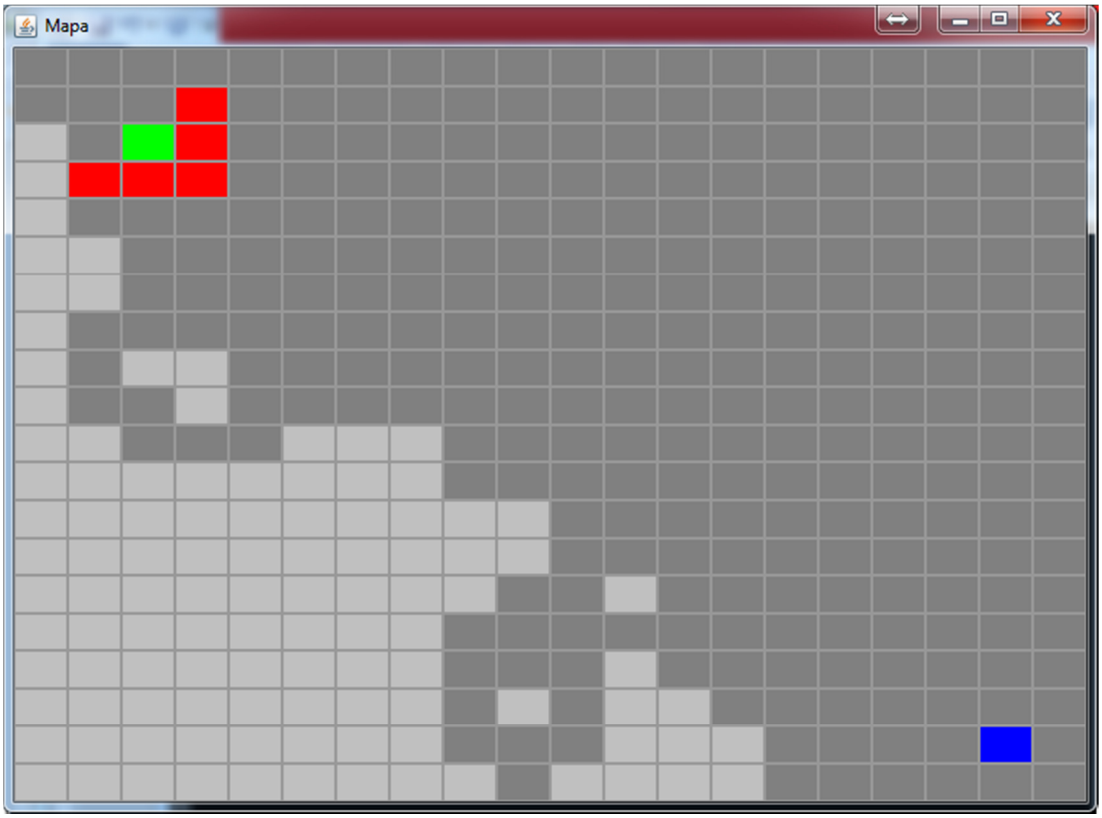


Figura 8- Resultado mapa fácil recorrido con gamma 0.5

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

Como puede observarse en la figura 9 el número de ciclos que recorre el programa es bastante más elevado que en los dos casos anteriores. Dado que el programa tiene menos en cuenta los refuerzos futuros la exploración del mapa es normal, no obstante puede observarse que se estabiliza pasados más o menos el mismo número de ciclos que en los casos anteriores. Esto es debido de nuevo a la disminución del parámetro épsilon que actúa de la misma manera que en los casos anteriores, tendiendo a estabilizar el número de ciclos recorridos.

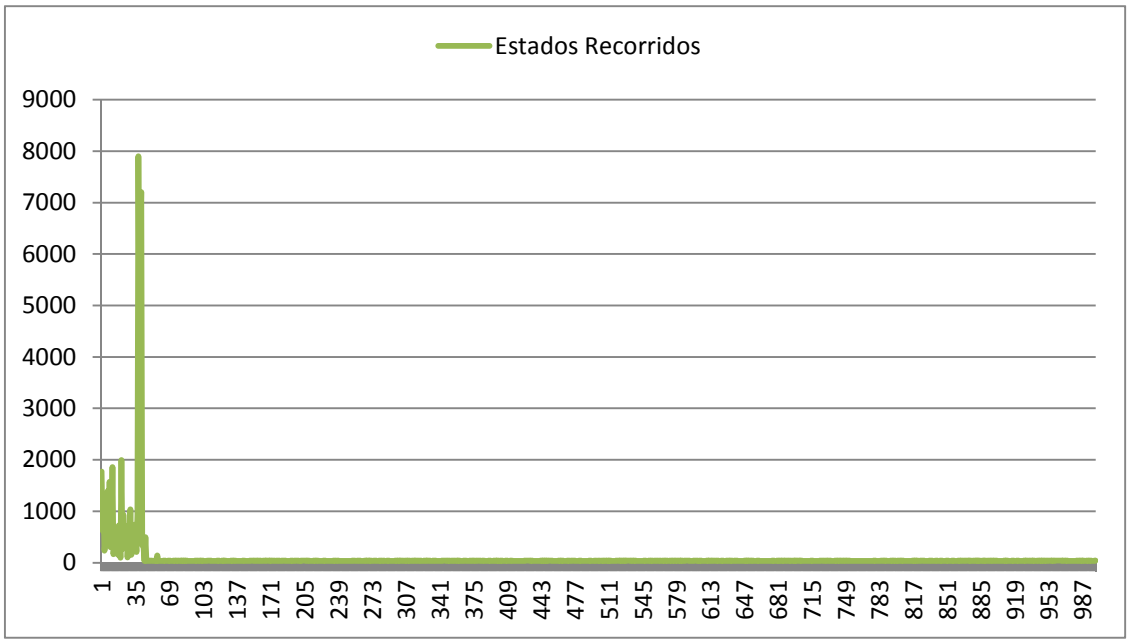


Figura 9 - Número de estados recorridos por ciclo en mapa fácil con gamma 0.5

5.2. Caso 2 - Medio

El caso número dos tiene está distribución de tesoro y de enemigos mostrada en la figura 10. Se considera de dificultad media debido a que el número de enemigos respecto al número de casillas aumenta hasta el 5%, no obstante este nivel de dificultad se obtiene por la distribución agrupada en una única zona.

Se toma como medida 10000 ciclos debido a que es un mapa de dificultad media y son necesarios más ciclos para resolverlo.

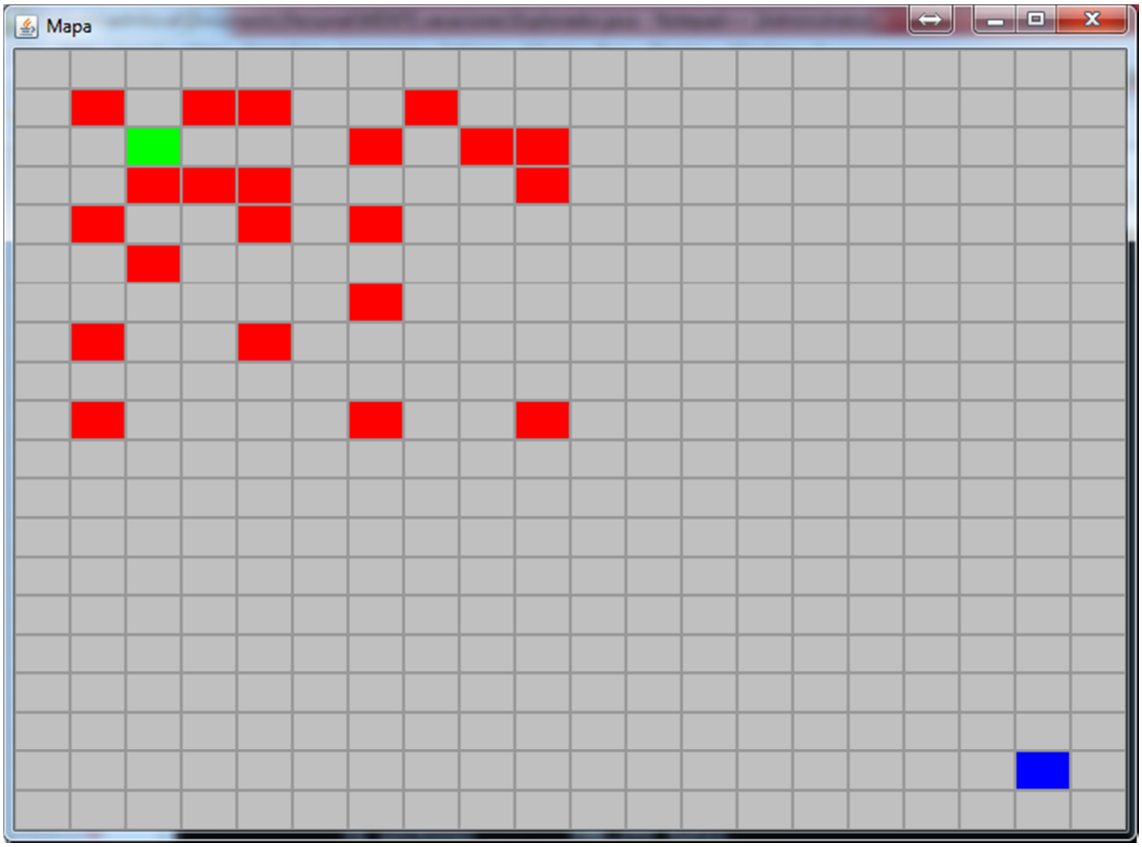


Figura 10 – Mapa dificultad media

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

5.2.1. GAMMA 0.9

El valor de gamma hace que el programa tenga bastante en cuenta el valor de los futuros refuerzos. No obstante puede observarse como el programa recorre practicamente el mapa completo (salvo un 2%), esto es debido a que el tesoro es más complicado de encontrar lo que provoca que el algoritmo tenga un valor de épsilon elevado durante mas ciclos.

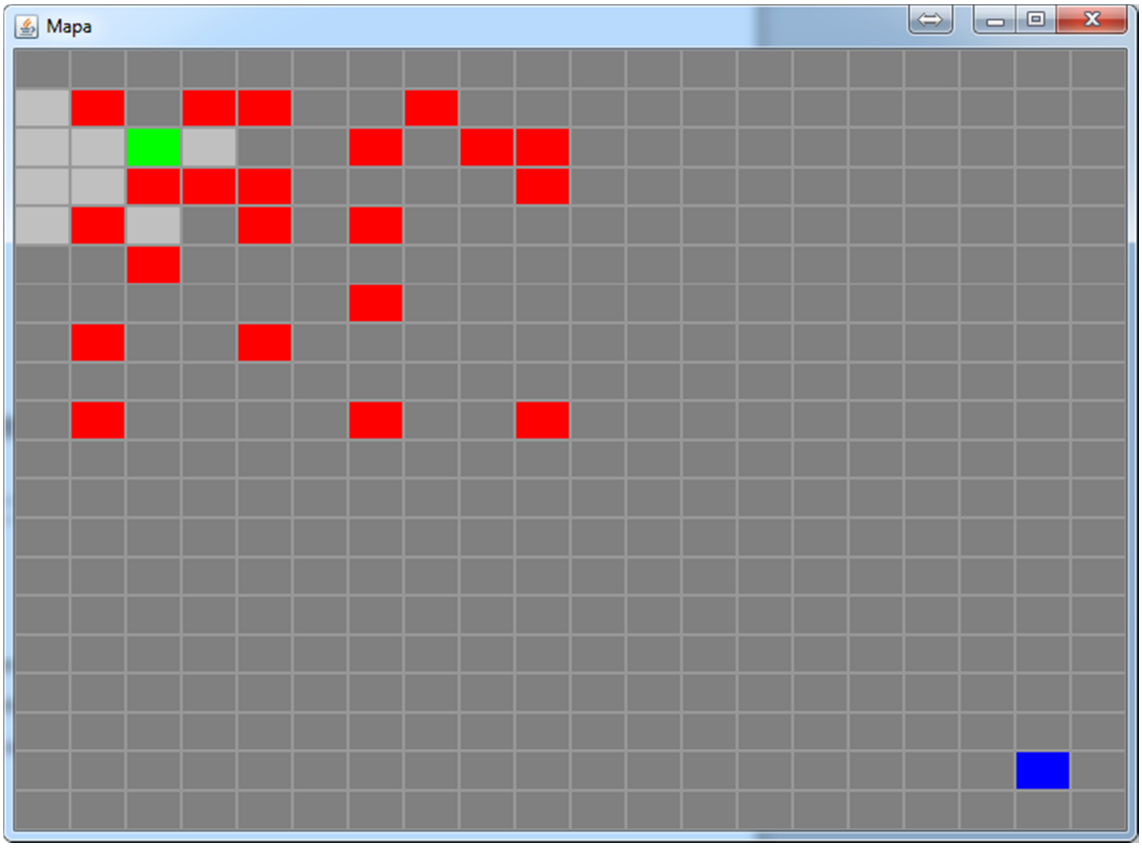


Figura 11 - Resultado mapa medio recorrido con gamma 0.9

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

La gráfica mostrada en la figura 12, es un claro ejemplo de lo descrito anteriormente, el número de estados recorridos por ciclo se mantiene bastante alto, hasta que tiende a estabilizarse alrededor del ciclo número 1.000 lo que indica que hasta entonces el valor de épsilon no se ha reducido lo suficiente para que la explotación se empiece a tener más en cuenta que la exploración.

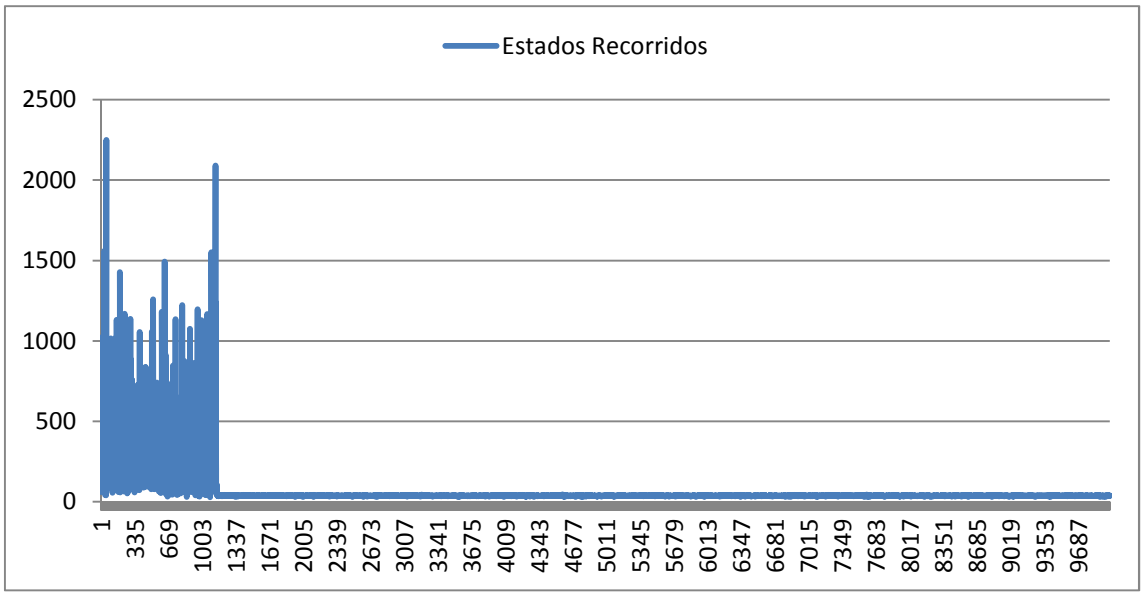


Figura 12 - Número de estados recorridos por ciclo en mapa medio con gamma 0.9

5.2.2. GAMMA 0.7

Con este valor de gamma el mapa explorado puede verse en la figura 13. En este caso al tener un poco menos en cuenta el valor de las recompensas futuras provoca que explore un poco más la zona llena de enemigos. Salvo dos casillas (0,5%) todas las demás son exploradas en algún momento. El valor alto épsilon hace que se explore un poco más al principio consiguiendo que prácticamente se explore todo el mapa.

Figura 13 - Resultado mapa medio recorrido con gamma 0.7

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

En la figura 14 puede observarse la gráfica obtenida de los resultados, ocurre como en el caso anterior el número de estados por ciclo tarda en reducirse, pero en este caso lo hace antes sobre el ciclo 700. Esto es debido a que con este valor de la variable gamma se tiene menos en cuenta el valor de recompensas futuras, por lo que encontrar antes el tesoro y por lo tanto reduce antes el épsilon que en el caso anterior.

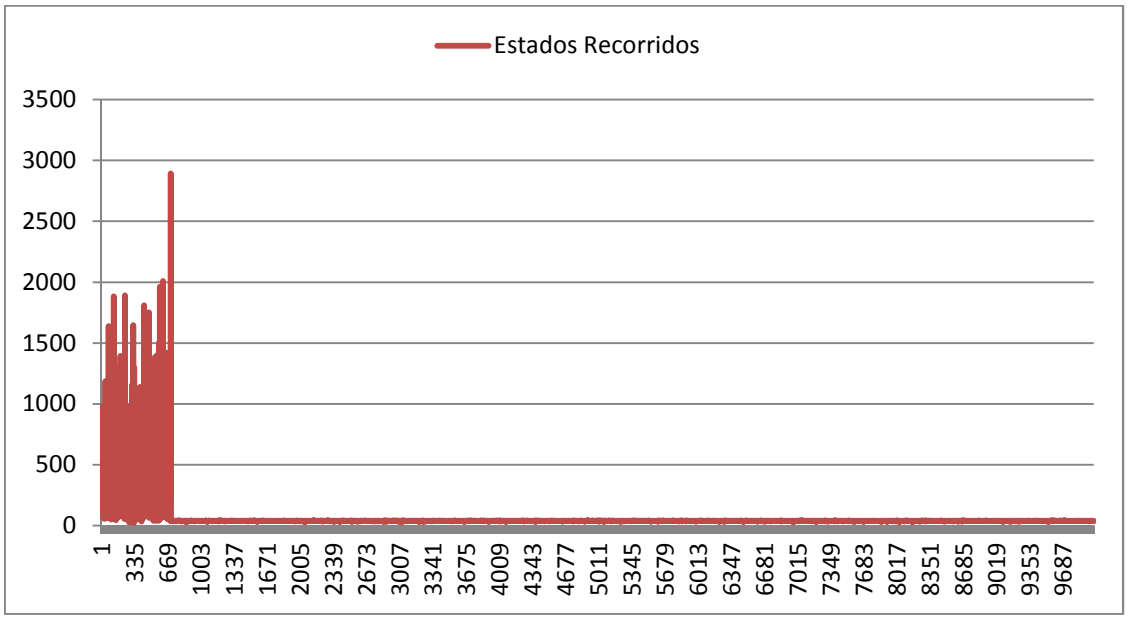


Figura 14 - Número de estados recorridos por ciclo en mapa medio con gamma 0.7

5.2.3. GAMMA 0.5

En la figura 15 se puede observar el resultado de este valor gamma, el algoritmo debe tener menos en cuenta el valor de los refuerzo futuros, el mapa está menos explorado que en los dos casos anteriores (no se explora un 11%) pudiera no parecer que se comporta correctamente, no obstante lo hace porque encuentra antes el tesoro, al no tener tanto en cuenta los refuerzos futuros decide explorar más casillas. De esta manera el valor de épsilon se reduce pronto reduciendo la aleatoriedad, lo que permite encontrar el tesoro antes.

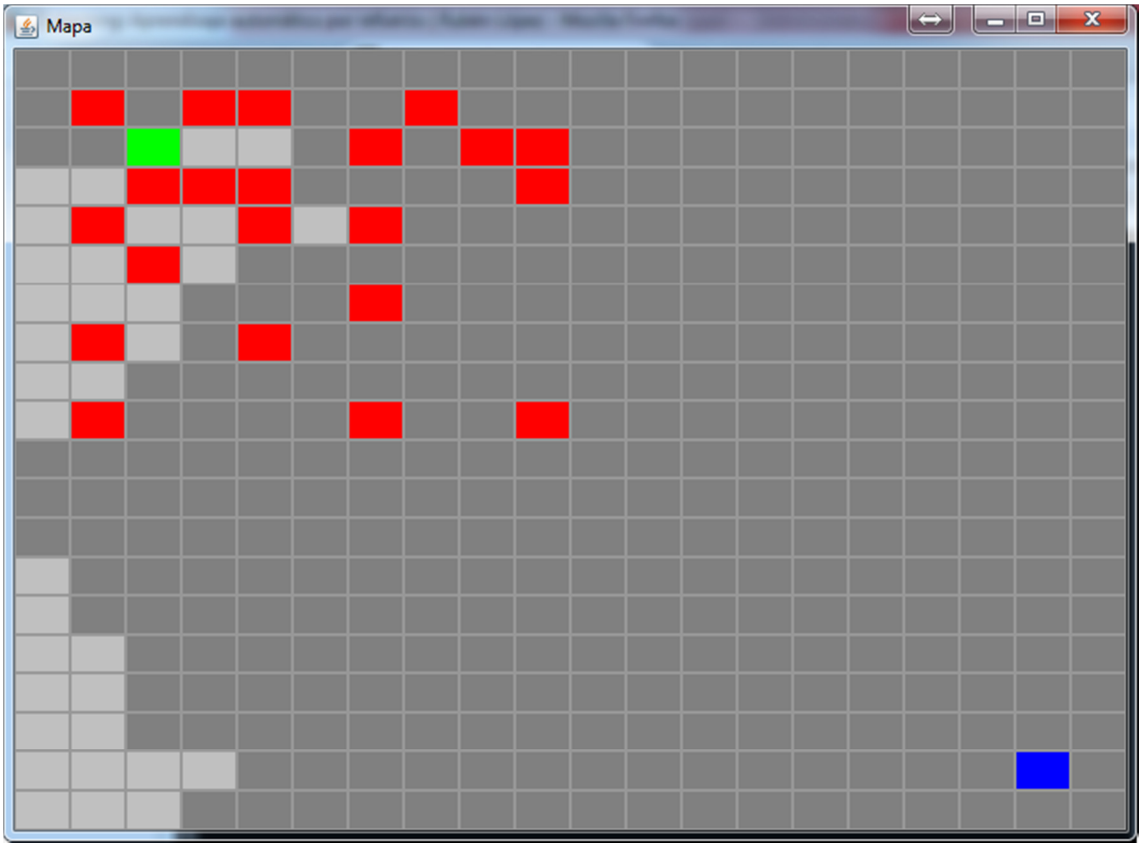


Figura 15 - Resultado mapa medio recorrido con gamma 0.5

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

En la figura 16 se muestra la gráfica que indica en cada ciclo el número de estados recorridos por ciclo. Dicho número se estabiliza en torno al ciclo 650 siendo esta estabilización un poco antes que en los dos casos anteriores, lo que corrobora que el tesoro es encontrado antes y el valor de épsilon es reducido antes.

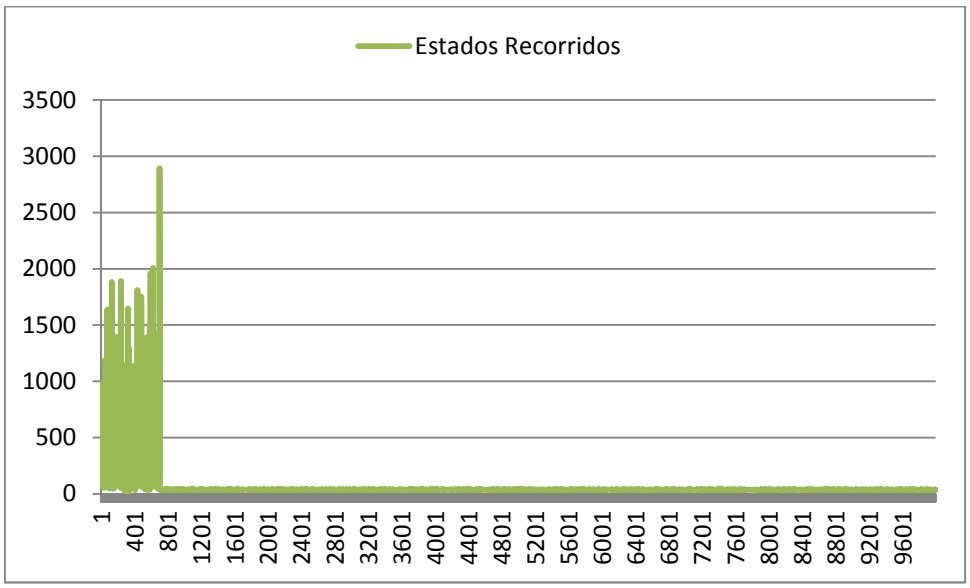


Figura 16 - Número de estados recorridos por ciclo en mapa medio con gamma 0.5

5.3. Caso 3 - Difícil

El caso número tres tiene esta distribución de tesoro y de enemigos como se muestra en la figura 17. Se considera difícil debido al gran número de enemigos en relación al número de casillas y la distribución en el mapa ocupando casi toda la extensión.

Ha sido necesario programar 500.000 ciclos para que pudiera encontrar el tesoro suficientes veces para sacar datos concluyentes. Es decir con este mapa si se programan 200.000 o 300.000 el programa es incapaz de encontrar el tesoro debido a que el valor de épsilon es alto y nunca es capaz de llegar a reducirlo un 10%. No se ha bajado el valor de épsilon porque todas las pruebas anteriores se han comenzado con un épsilon al 90%.

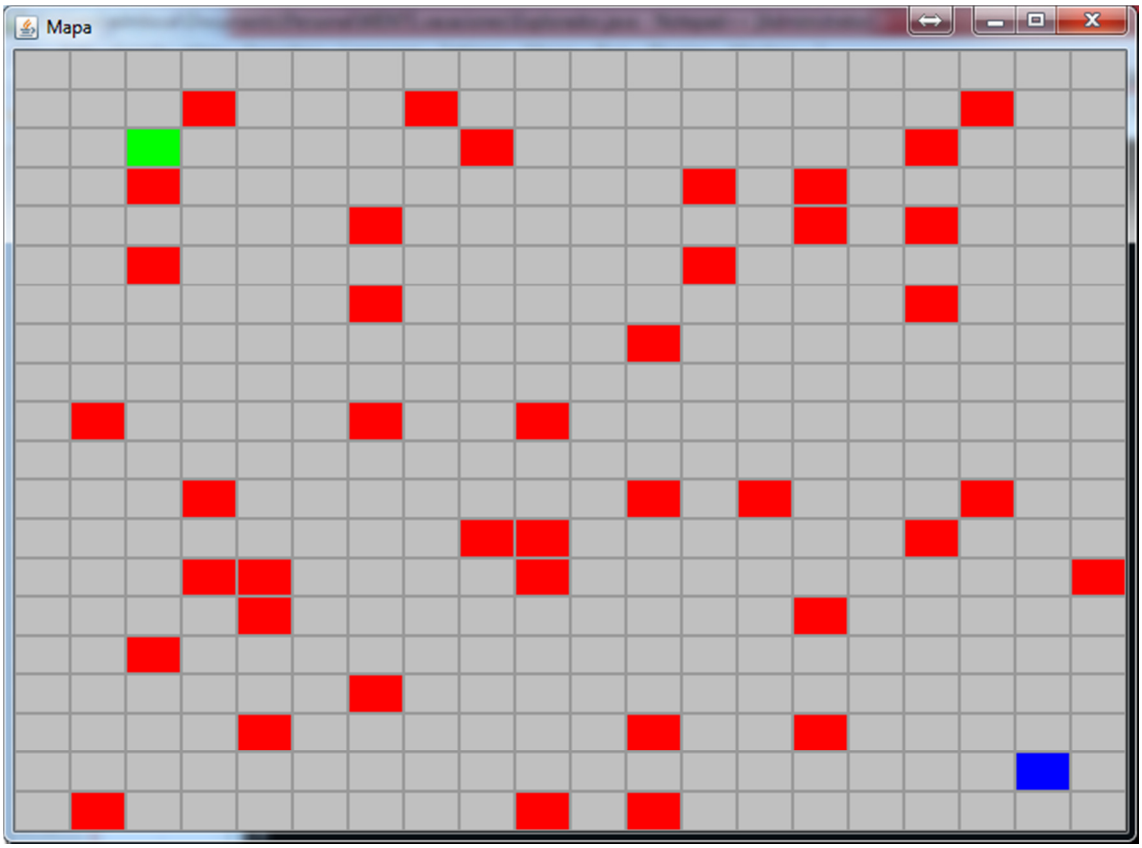


Figura 17 – Mapa dificultad difícil

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

5.3.1. GAMMA 0.9

El valor de gamma alto hace que el algoritmo tenga más en cuenta los refuerzos futuros, salvo un 2% el mapa es explorado completamente, este 2% es exactamente el mismo valor de casillas no exploradas que el caso 2 por lo que tiende a comportase de una manera parecida. El tesoro es muy complicado de encontrar lo que provoca que el algoritmo tenga un valor de épsilon elevado durante muchos ciclos.

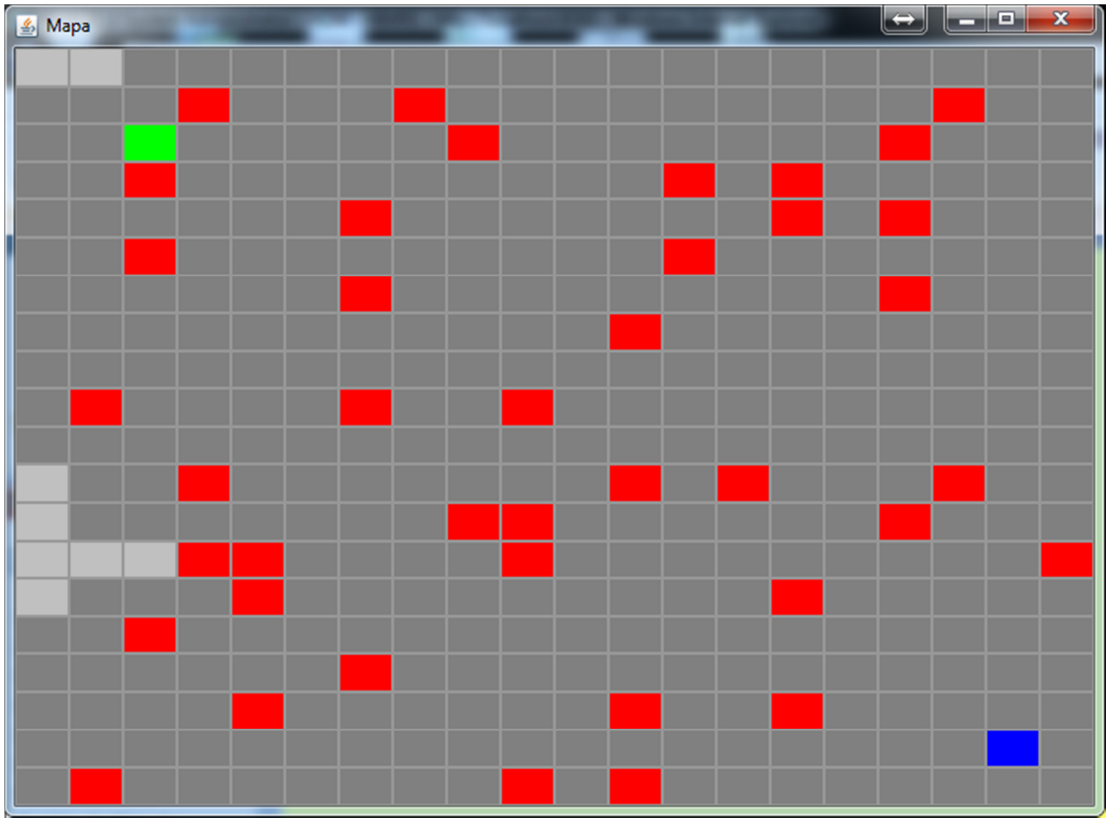


Figura 18 - Resultado mapa difícil recorrido con gamma 0.9

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

La gráfica mostrada en la figura 19 representa el número de estados recorridos por ciclo, dicho valor se mantiene bastante alto hasta que tiende a estabilizarse alrededor del ciclo número 435.000 lo que indica que hasta entonces el valor de épsilon no se ha reducido lo suficiente para que se tenga más en cuenta la explotación que la exploración, exactamente como en el caso 2.

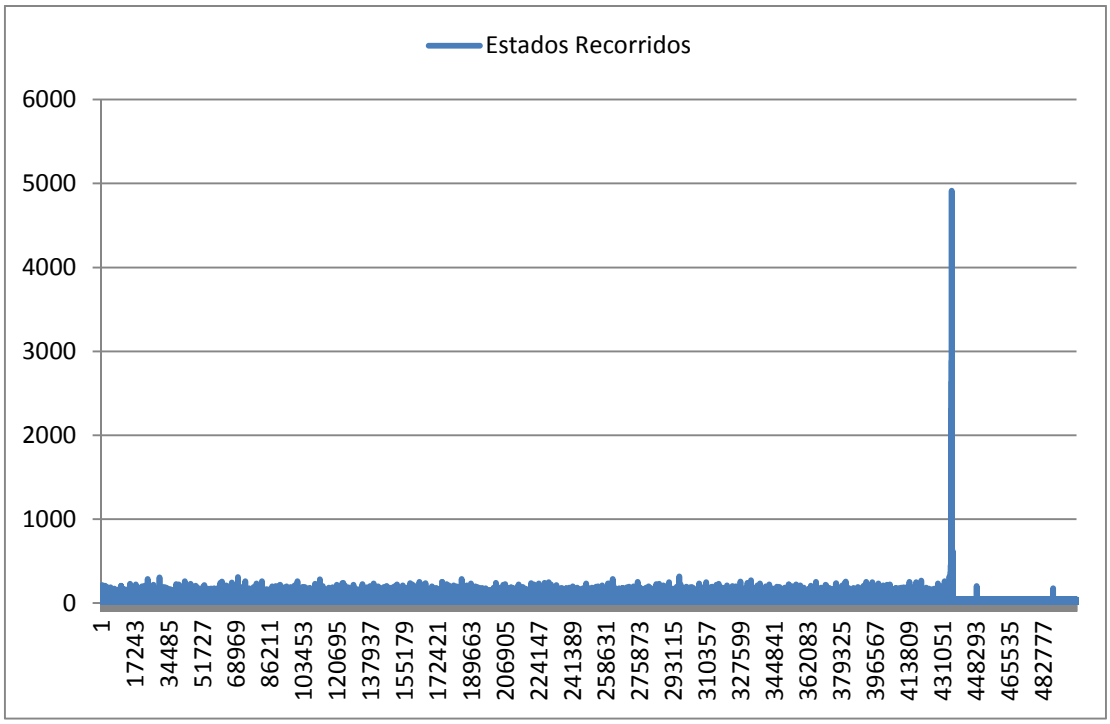


Figura 19 - Número de estados recorridos por ciclo en mapa difícil con gamma 0.9

5.3.2. GAMMA 0.7

Con este valor de gamma el mapa explorado puede verse en la figura 20, en este caso de nuevo la pequeña variación en la variable hace que tenga un poco más en cuenta el valor de las recompensas futuras, el mapa esta menos explorado de hecho se queda sin explorar un 9,5% de las casillas. Todas las demás son exploradas en algún momento. El programa encuentra el tesoro un poco antes de esta manera el valor de épsilon alto se reduce en menos ciclos.

Figura 20 - Resultado mapa difícil recorrido con gamma 0.7

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

La grafica mostrada en la figura 21 representa el número de estados recorridos por ciclo. Dicho valor se mantiene bastante alto hasta que tiende a estabilizarse alrededor del ciclo número 340.000. Al tener menos en cuenta los valores de futuros refuerzos encuentra antes la solución.

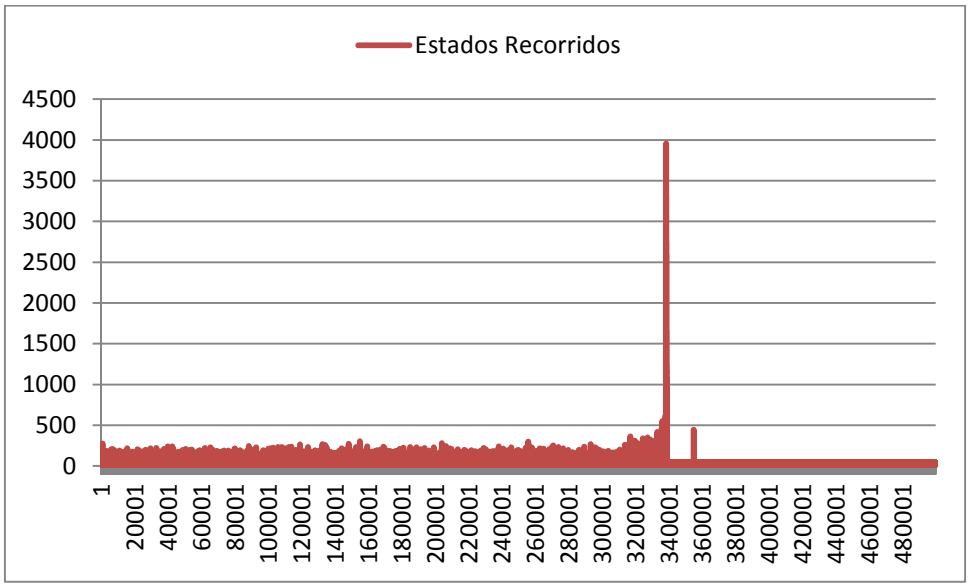


Figura 21 Número de estados recorridos por ciclo en mapa difícil con gamma 0.7

5.3.3. GAMMA 0.5

En la figura 22 se puede observar el resultado de este valor gamma, el programa tiene menos en cuenta el valor de los refuerzos futuros, el mapa esta menos explorado que en los dos casos anteriores (no se explora un 12,5%), de nuevo pudiera no parecer que se comporta correctamente como en el caso número 2. No obstante lo hace porque encuentra antes el tesoro. De esta manera el valor de épsilon se reduce pronto reduciendo la aleatoriedad, lo que permite encontrar el tesoro antes.

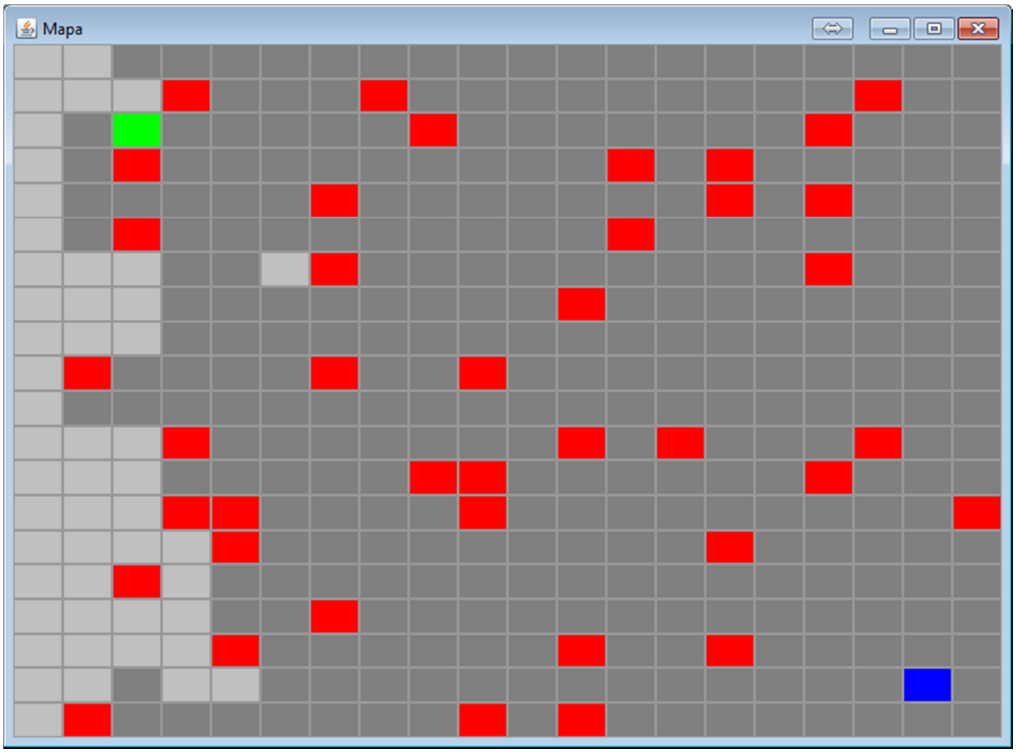


Figura 22 - Resultado mapa difícil recorrido con gamma 0.5

LEYENDA:
Explorador
Enemigo
Tesoro
Casilla sin explorar
Casilla Explorada

En la figura 23 se muestra la gráfica que indica en cada ciclo el número de estados recorridos por ciclo. Dicho número se estabiliza en torno al ciclo 55.000 produciéndose esta estabilización mucho antes que en los dos casos anteriores, lo que corrobora que el tesoro es encontrado antes y el valor de épsilon es reducido antes.

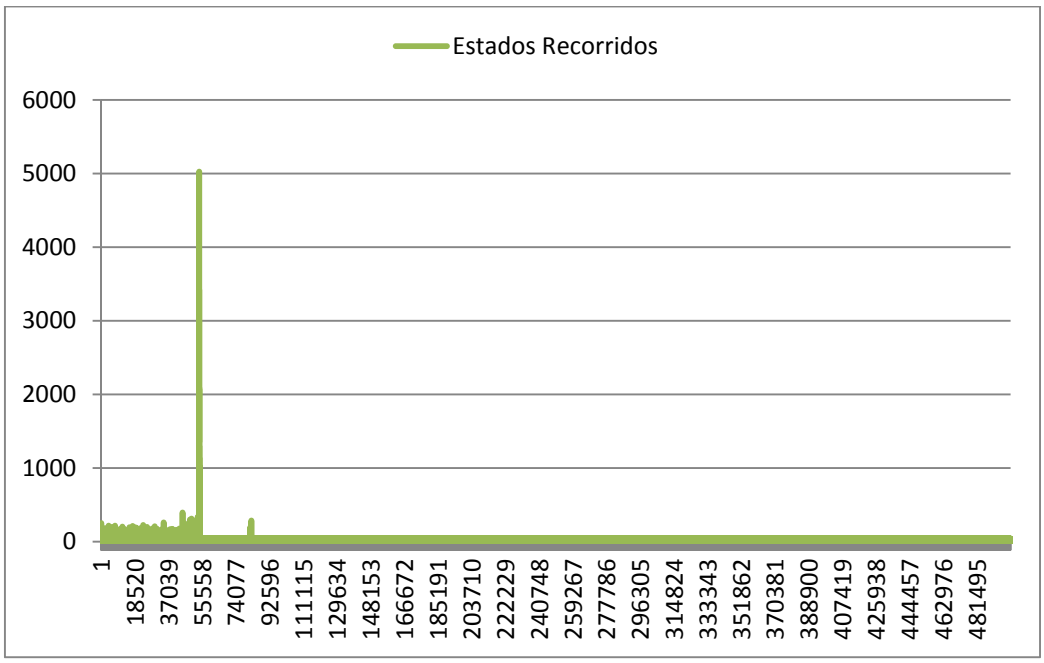


Figura 23 Número de estados recorridos por ciclo en mapa difícil con gamma 0.5

6. RESULTADOS

En esta sección expondré los resultados analizados de cada uno de los tres casos, está estructurado de tal manera que las gráficas muestran los 3 valores de gamma de un mismo caso, a fin de poder compararlos más gráficamente. El análisis de estos resultados me permitirá establecer cuál es valor óptimo de gamma en cada uno de los casos.

Dado que este PFC se basa en realizar una experimentación con la variable gamma asociada al concepto de dificultad, no se ha tenido en cuenta el rendimiento ni la optimización del algoritmo Q-learning, motivo por el cual no he tomado tiempos para comparar con otros algoritmos ni se han tenido en cuenta el número de aciertos o veces que el algoritmo encuentra el tesoro o muere. Estos sucesos son indiferentes para analizar los casos planteados.

Estos resultados se han analizado desde el punto de vista de número de estados recorridos por ciclo, además he tenido en cuenta el número de ciclo en el que se empieza a estabilizar este número, ya que cuando empieza a estabilizarse es porque el algoritmo ha encontrado el camino y lo repite constantemente.

De esta manera se puede analizar cómo es de bueno un valor gamma para cada caso, hasta que se produce el efecto de conocer el camino por dónde se debe avanzar, permitiendo así las conclusiones expuestas en la sección 7.

6.1. Caso 1 resultados detallados

En la figura 24 y 25 puede observarse una gráfica más detalla del caso 1, comparando los 3 valores de gamma más en detalle.

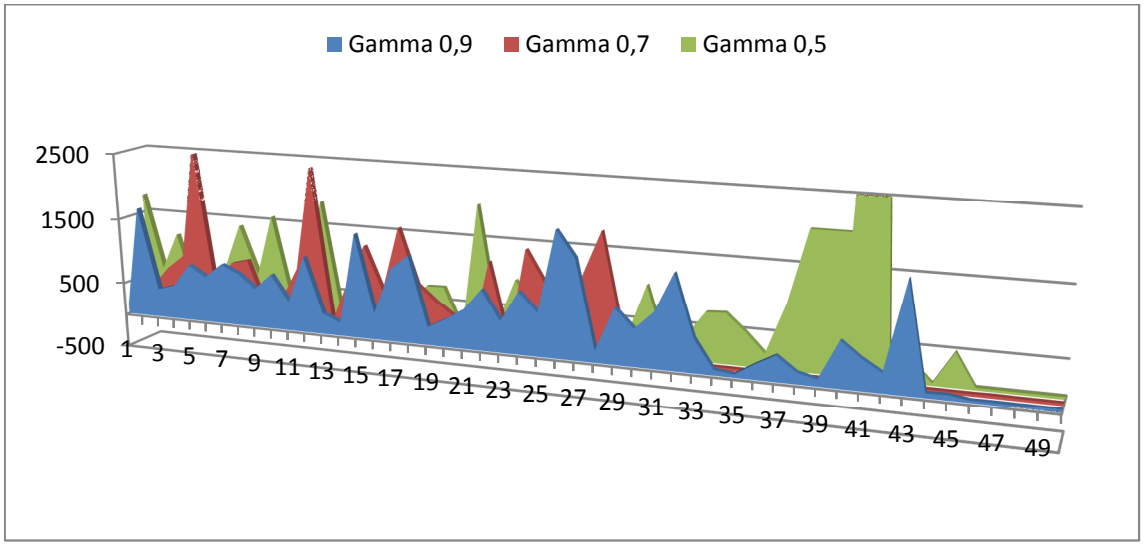


Figura 24

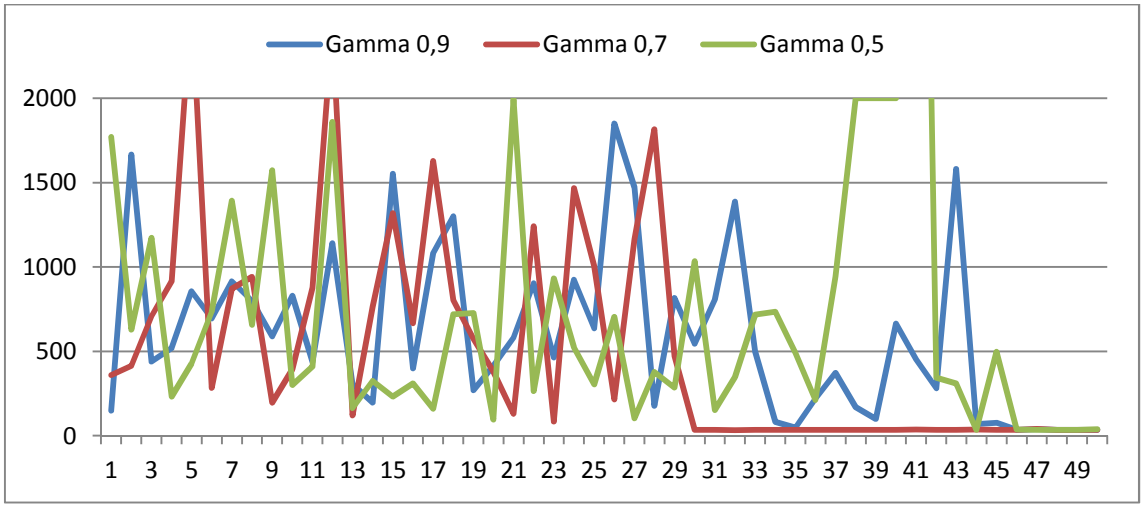


Figura 25

Número medio de ciclos:

Promedio Gamma 0,9: 598

Promedio Gamma 0,7: **511 Mejor promedio**

Promedio Gamma 0,5: 771

Como se observa claramente en la gráfica para cada uno de los valores de gamma en los primeros 25 ciclos se producen alteraciones bastante marcadas en el número de estados recorridos por ciclos. Estos picos en las gráficas son normales, debido a que en los primeros ciclos la variable épsilon tiene un valor muy alto casi un 90% de aleatoriedad, por lo que apenas se muestran distinciones entre un valor de gamma y otro.

Es a partir de ciclo 25 en el que se observan claras diferencias entre unos valores de gamma y otros, estos es debido a que el valor gamma puede traducirse como la importancia que se le otorga a los refuerzos futuros recibidos.

El valor 0,5 resulta ser el peor de todos, ya que al tener tan en cuenta los valores que sabe, pierde opciones de explorar determinados movimientos que le permitirían aprender más y reducir el número de estados recorridos por ciclos. El valor 0,9 es un valor más standard y más común para estos algoritmos, evita el problema que tiene el gamma anterior, pero a costa de tardar más en encontrar el objetivo por no tener en cuenta tanto todo de lo que ha aprendido hasta ese momento. El valor 0,7 es ideal en este caso como muestra la gráfica, es el más rápido pues tiene en cuenta a partes iguales la exploración y la explotación.

Por lo tanto para este caso el mejor valor que he probado resulta ser ***gamma 0,7***

6.2. Caso 2 resultados detallados

A continuación en la figura 26 se muestra una gráfica más detallada del caso 2, además en la figura 27 se muestra una sección ampliada de la figura 26. En ellas se comparan los 3 valores de gamma. La grafica se corta cuando los tres casos tardan el mismo numero de ciclos en concluir y no se extienden hasta 10.000 que es el numero de ciclos maximo.

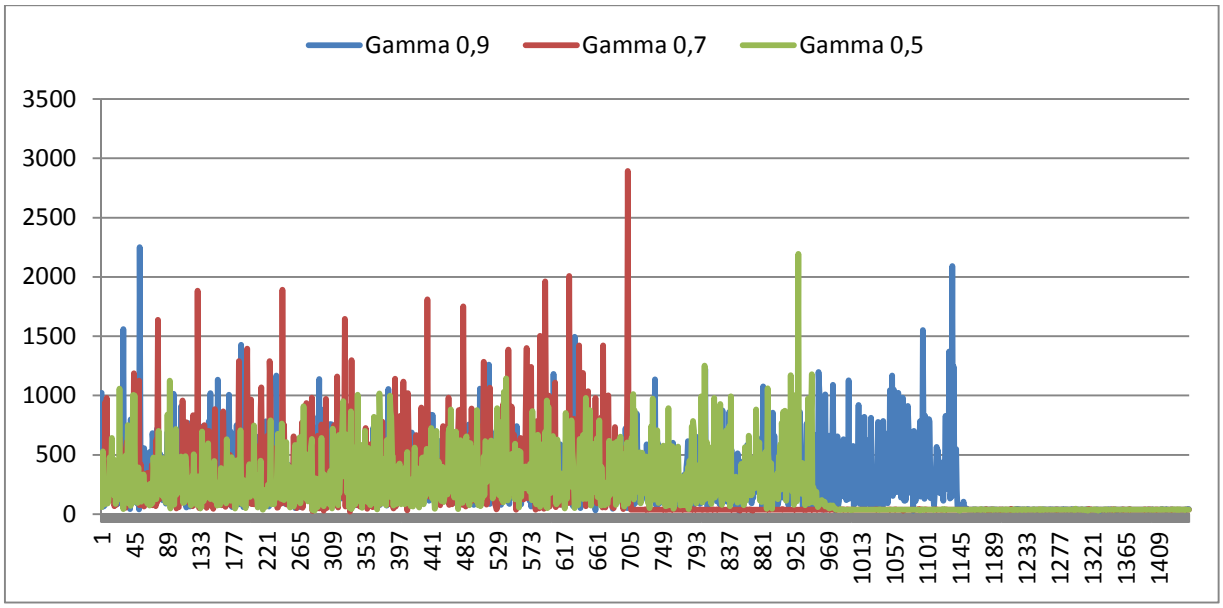


Figura 26

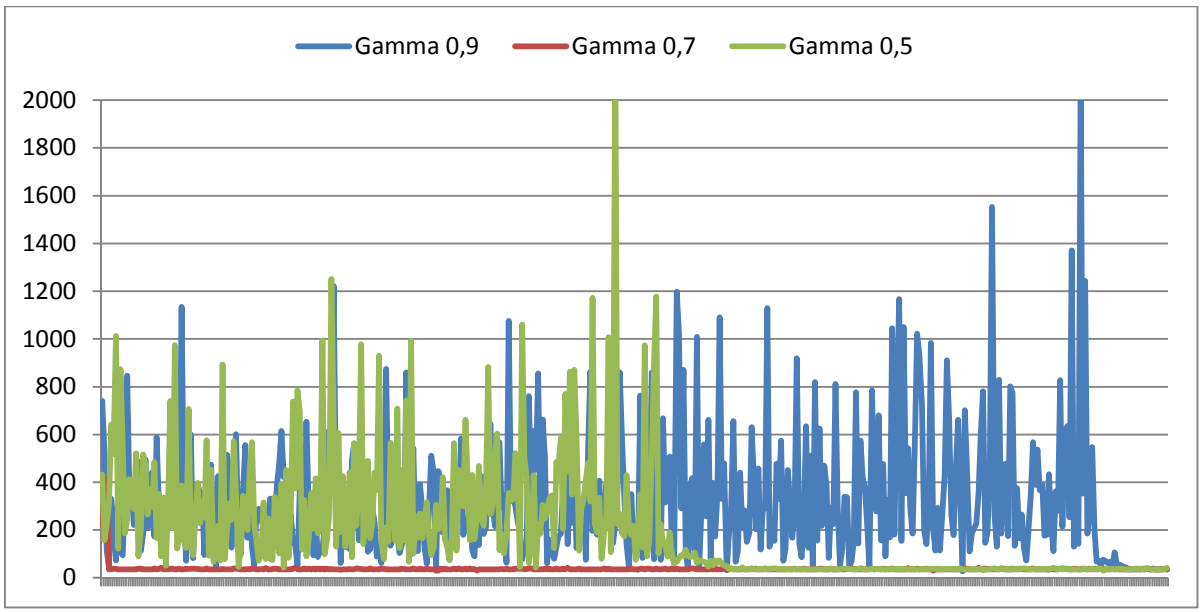


Figura 27

Como puede observarse en las gráficas, el número de estados recorridos en cada ciclo para los 3 valores de gamma, fluctúa hasta que el número de ciclos que se han realizado tomo un valor relativamente alto. Prestando atención al detalle ampliado a la segunda gráfica, se pueden observar estos resultados.

El valor 0,9 es el que peor comportamiento tiene ya que debido a ser tan precavido, obliga a recorrer muchas partes del mapa y dado que el mapa esta tan poblado de enemigos, tarda muchísimo tiempo en aprender el camino.

Al valor 0,5 le ocurre lo mismo en que en el caso anterior, ya que al tener tan en cuenta los valores que conoce, pierde opciones de explorar determinados movimientos que le permitirían aprender más y reducir el número de estados recorridos por ciclos, a pesar de ser un mapa tan "poblado" aprender un poco más le ayudaría a poder encontrar antes el camino óptimo hacia el tesoro.

El valor 0,7 de nuevo muestra el mejor comportamiento de los tres, en este caso como muestra la gráfica, es el más rápido en encontrar un camino estable, ya que como ya se citó pues tiene en cuenta a partes iguales la exploración y la explotación.

De nuevo el mejor valor que he probado resulta ser ***gamma 0,7***

6.3. Caso 3 resultados detallados

En las figuras 28, 29 y 30 se muestran las gráficas del caso 3 para los distintos valores de gamma.

Gamma 0,9

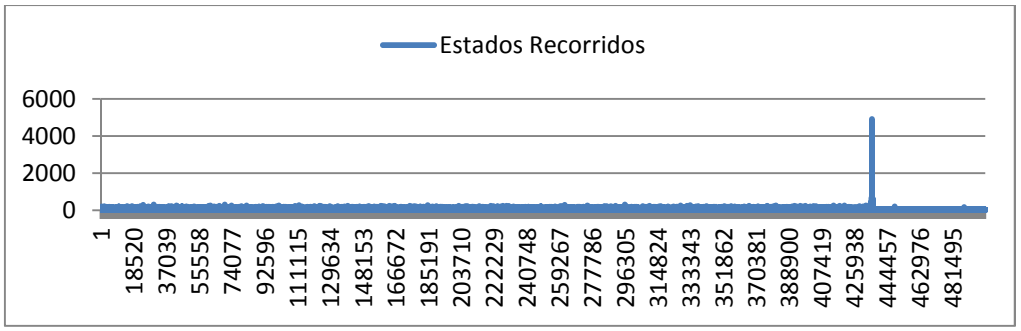


Figura 28

Gamma 0,7

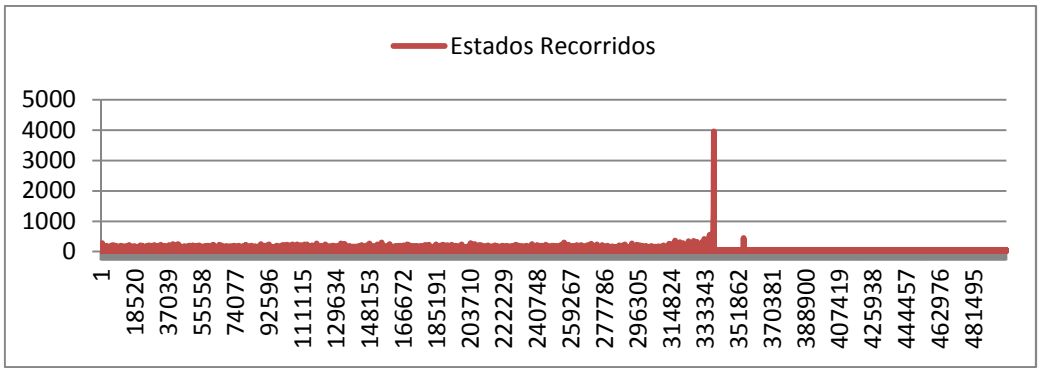


Figura 29

Gamma 0,5

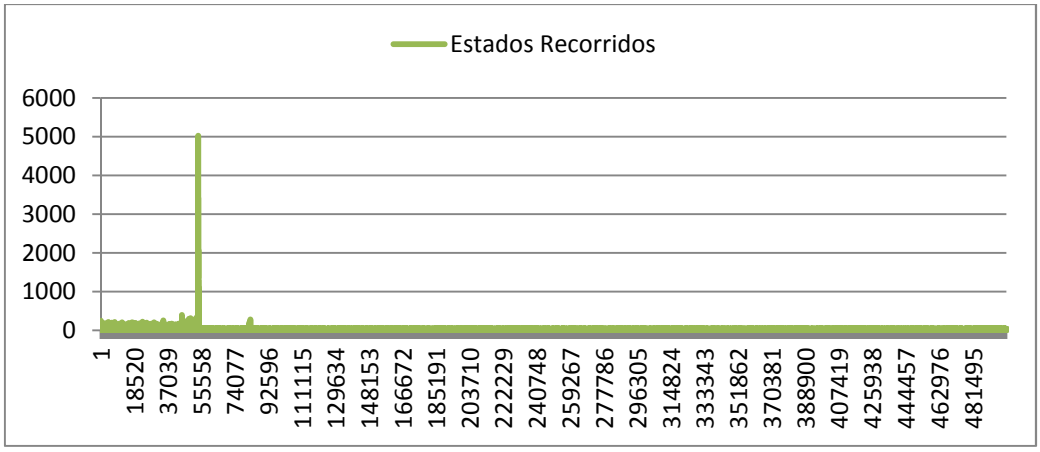


Figura 30

En este caso de dificultad llamado difícil, no es tan importante el comienzo y la fluctuación inicial, si no el momento en el que se estabilizan el número de ciclos, en las gráficas conjuntas no se aprecia mucha variación en el número de estados recorrido entre el lado izquierdo “del pico” y el lado derecho, pero este valor es bastante menor.

Dado la enorme cantidad de ciclos (500.000) que han sido necesarios para encontrar el camino correcto suficientemente número de veces y el altísimo número de estados recorridos en determinados ciclos, la gráfica no es visualmente tan efectiva como debería.

Los picos que se observan se producen cuando está cerca de encontrar el camino correcto y recorre el mapa en un único ciclo, recorriendo multitud de estados buscando la siguiente acción correcta para encontrar el tesoro, sabiendo bastante bien donde están los enemigos.

En este caso poco se puede decir del comportamiento de los distintos valores de gamma por separado, como puede verse claramente el mejor valor de gamma para este caso es de 0,5 ya que es el valor que primero se estabiliza.

Así que por primera el mejor valor que he probado resulta ser ***gamma 0,5***

7. CONCLUSIONES

Tras analizar los datos del apartado anterior se pueden extraer diversas conclusiones de este PFC. A continuación se enumeran las que a mi entender son las conclusiones más claras y concretas.

- Los valores altos de la variable gamma no garantizan en ninguno de los tres casos ninguna ventaja específica más que en términos de exploración. Ya que se explora más extensamente del mapa, permitiendo que el algoritmo sea capaz de explorar más territorio, pero a costa de perder eficiencia ya que tarda más en encontrar la solución.

- Los valores medio altos de la variable gamma muestran el mejor rendimiento en términos de encontrar la solución antes. No obstante en mapas con dificultad alta, no logran el mismo rendimiento que un valor menor de gamma.

- La variable gamma influye mucho en la cantidad de mapa que se explora, para entornos de más de 40x40 casillas es probable que no se explore el mapa al completo, especialmente para valores bajos de gamma, y siempre y cuando la dificultad sea sencilla o moderada. Es decir influye directamente en el balance entre exploración y explotación.

La conclusión es que el valor de gamma no es un valor a fijar y utilizar para todos los casos con el mismo valor, con los resultados obtenidos queda bastante claro que su valor influye y mucho en el comportamiento del algoritmo.

Queda demostrada la complejidad de acertar este valor en un entorno determinista ya que cambiará mucho el resultado en función de su valor.

No puedo terminar estas conclusiones sin hacer notar que en los casos de estudio expuestos solo hay un refuerzo positivo, creo que si existieran muchos refuerzos positivos los valores bajos de gamma harían al algoritmo muy eficaz pero también harían que el valor de exploración fuera bajo, pasando por alto a otros refuerzos positivos, siendo así incapaz de explorar el mapa completamente sobre todo en entornos fáciles.

Por lo tanto como conclusión cabe decir que el mejor valor de esta variable gamma ha de ser necesariamente distinto para cada caso.

Mi conclusión final es que la variable gamma no debería ser siempre la misma, ni debería estar prefijada de antemano, debería ser ajustada teniendo en cuenta la información previa al inicio de la búsqueda siempre que esta información esté disponible, digamos que sería bueno ajustarla en función de la dificultad del caso al que nos enfrentemos, en caso de no estar disponible esta información se configurara el valor de ***gamma a 0,9*** que es el valor que tiene más en cuenta los valores del refuerzo futuro.

Incluso sería necesario ir cambiando su valor dinámicamente en función de lo que esté aprendiendo el algoritmo, según vaya descubriendo la dificultad del caso que esté explorando.

8. LINEAS FUTURAS

Este PFC puede ser continuado o ampliado por otras personas, está diseñado para que sea fácilmente modificable por lo tanto se puede continuar en varias líneas, a continuación muestro las que me han parecido más interesantes:

- El código fuente puede ser modificado para hacerlo no determinista y elaborar las mismas pruebas, analizando así el comportamiento de la variable gamma en otro entorno.

- Puede reutilizarse como una clase básica, parametrizando sus propiedades, para utilizarla como método de búsqueda en diversos mapas.

- Puede ampliarse, con métodos que le ayuden a determinar la dificultad de la exploración en curso, para así poder variar dinámicamente la variable gamma, y hacer un análisis observando el comportamiento de la misma, para ver si reduce así el tiempo en encontrar el camino correcto.

- Se puede introducir el concepto de dificultad en otros sistemas más complejos de búsqueda, para obtener resultados y ver cómo influye la dificultad en los distintos parámetros.

- Se puede hacer no determinista y estudiar el valor de la variable alfa y si influye en la gamma en determinados entornos de dificultad. Mediante la fórmula no determinista.

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha[r + \gamma \max_{a'} Q(s',a')]$$

Y existen multitud de líneas futuras debido a desarrollar.

9. BIBLIOGRAFIA

1. *Aprendizaje por refuerzo en espacios de estados continuos. Tesis Doctoral. Universidad Carlos III de Madrid. Fernando Fernández Rebollo, 2002.* http://e-archivo.uc3m.es/bitstream/handle/10016/569/Fernandez%20Rebollo,%20Fernando%281%29.pdf

2. *Reinforcement Learning: a Survey. L. P. Kaelbling, M. L. Littman y A. W. Moore. Journal of Artificial Intelligence Research 4. 1996.* https://www.jair.org/media/301/live-301-1562-jair.pdf

3. *Machine Learning. Tom. Mitchell. McGraw-Hill.1997. ISBN:* 0070428077 http://personal.disco.unimib.it/Vanneschi/McGrawHill_-_Machine_Learning_-Tom_Mitchell.pdf

4. *Reinforcement learning. An introduction R. Sutton, A. Barto Mit Press 1998.* http://neuro.bstu.by/ai/RL-3.pdf

5. *Aprendizaje automático: Fernando Fernández Rebollo y Daniel Borrajo. 2009* http://ocw.uc3m.es/ingenieria-informatica/aprendizaje-automatico/material-de-clase-1/aa-ocw-refuerzo.pdf

6. *Machine Learning 8. Christopher J.C.H. WATKIN y Peter Dayan. 1992 Kluwer Academic Publishers.* http://www.gatsby.ucl.ac.uk/~dayan/papers/cjch.pdf

7. *Learning Rates for Q-learning. Eyal Even-Daartlett y Yishay Mansour Journal of Machine Learning research. 2003.* http://www.jmlr.org/papers/volume5/evendar03a/evendar03a.pdf

8. *Bayesian Q-learning. R. Dearden. Nir Friedman y Stuart Rusell. AAI organization. 1998.* http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.301.6263&rep=rep1&type=pdf